HOTEL BOOKING DEMAND PREDICTION USING MACHINE LEARNING MODELS

A project report submitted in partial fulfilment
of the requirements for the award of degree in

M.Sc. Data Science

Submitted by
Sneha J. Vora
Regd. No: VP21CSCI0200081

Under the esteemed guidance of

Mr. M. Suresh Kumar
Assistant Professor

Department of Computer Science
GITAM School of Science
GITAM (Deemed to be University)
Visakhapatnam – 530045, A.P

(2022-2023)

CERTIFICATE

This is to certify that the project entitled **“HOTEL BOOKING DEMAND PREDICTION USING MACHINE LEARNING MODELS”** is a bonafide work done by **Sneha J. Vora, Regd. No: VP21CSCI0200081** during **December 2022 to April 2023** in partial fulfillment of the requirement for the award of degree of **M.Sc. Data Science** in the Department of Computer Science, GITAM School of Science, GITAM (Deemed to be University), Visakhapatnam.

Internal Guide	**Head of the Department**
Mr. M. Suresh Kumar	Dr. T. Uma Devi
Assistant Professor	Associate Professor
Department of Computer Science	Department of Computer Science
	Gitam School of Science
	GITAM

DECLARATION

I, **Sneha J. Vora, Regd. No: VP21CSCI0200081** hereby declare that the project entitled **“HOTEL BOOKING DEMAND PREDICTION USING MACHINE LEARNING MODELS”** is an original work done in the partial fulfillment of the requirements for the award of degree of **M.Sc. Data Science** in GITAM School of Science, GITAM (Deemed to be University), Visakhapatnam. I assure that this project work has not been submitted towards any other degree or diploma in any other colleges or universities.

Sneha J. Vora

Regd. No: VP21CSCI0200081

ACKNOWLEDGEMENT

I would like to express profound gratitude to Mr. M. Suresh Kumar, Assistant Professor for giving me guidance, suggestions, thought provoking discussion, constant encouragement and allowing me the opportunity to work on this project.

I am extremely thankful to our project in charge V. Satyasaivani, Associate Professor who guided me throughout the project. Her valuable guidance and wise approach helped me to bring out this project successfully.

I am very much grateful to our Head of the Department Dr. T. Uma Devi, Associate Professor and faculty member of our department for her valuable coordination and guidance which helped me to bring this project successfully.

I am very much thankful to our beloved Principal Prof. K. Vedavathi for her kind support and facilities provided at our campus which helped me to bring out this project successfully.

I express my gratitude to our beloved Dean Prof. Balkumar Marthi, for his invaluable support and guidance provided by him throughout the course.

Finally, I would like to convey my heart full thanks to all Technical Staff, for their guidance and support in every step of this project. I convey my sincere thanks to all the faculty and friends who directly or indirectly helped me for the successful completion of this project.

Sneha J. Vora

Regd. No: VP21CSCI0200081

INDEX

ABSTRACT

Hotel Booking Demand Prediction using Machine Learning Models is an accurate booking cancellation forecast by which a user can know the things related to hotel bookings very easily. Booking cancellation has a significant effect on revenue of the business. To reduce the cancellation effect, the hotels can apply the cancellation model as a key to addressing this problem with the machine learning-based system developed. By combining data science tools and capabilities with human judgement and interpretation, this project aims to demonstrate how the predictive analysis of the model can contribute to synthesizing and predict about booking cancellation forecasting. Furthermore, this project aims, by detailing the full prediction & analysis, to give relaxation to user who wants to apply in particular hotel. The prediction model enables the hotel managers to mitigate revenue loss derived from booking cancellations and to mitigate the risks associated with overbooking (reallocation costs, cash or service compensations). By implementing various algorithms like Logistic Regression, K Nearest Neighbors (KNN), Random Forest, Decision Tree, etc. to classify the data and also use Evaluation Matrix to separate categorical data in a particular type, user can know the prediction up to the desired level.

1. INTRODUCTION

With the ever expanding use of the internet and online travel booking sites for vacation planning, opportunities may exist for niche travel reservation offerings in the Online Travel Agency industry. Hotels play an important role for tourists whether the tourist is local or international. They provide a lot of services to their customers such as parking area, food, room service and a lot more. By providing these services Hotels take the valuable feedback from their customers. These feedbacks help them in maintaining their reputation in the city/area. If the services are poor, the bookings of that hotel are low and if the services are awesome then high bookings in that hotel takes place.

In this model, the possibility of a booking for a hotel is predicted based on different factors and if they need special requests based on the factors. In this project the dataset used contains both international and local hotel data. Popular Machine Learning Algorithms like Decision Tree, Random Forest, KNN, Logistic Regression, etc. to predict the cancellation chances. This Model gives the prediction of Hotel Booking Cancellation up to the certain level of accuracy i.e. 95%. These machine learning techniques involve identifying similar customers based on their purchase history and behavior. By analyzing similarities between customers, the algorithm can generate product recommendations that are tailored to each individual's preferences.

On the other hand, machine learning techniques involve breaking down data into smaller, more manageable subsets and making decisions based on those subsets. The algorithms also consider customer preferences when making predictions. By analyzing a customer's booking history and browsing behavior, the algorithm can identify patterns of the booking being cancelled or successful.

By providing such kind of hotel demand and cancellation predictions, these machine learning models help the hotel owners in optimizing their business towards higher profitability.

1.1. Background

Booking cancellation is a very common thing in today’s world, which can cause severe losses to the business owners. This paper describes how AI is used to identify which booking can be cancelled and prevent some losses. The machine learning model should be evaluated in the real time environment for accuracy. Prediction model of hotel booking cancellation no doubt the issue that can be resolved in the context of Design Science Research (DSR), as it need to develop an artifact, here in this particular case, a form of Revenue Management System (RMS), fulfilling the two requirements of DSR.

The growing trend of Hotels Industry is beneficial for Hotels but there are some problems too such as Rising Rate of Cancellation. The user cancels the booking of the hotel after seeing the reviews given by the people who booked the hotel already.
In Some case hotel owners treat the customer in bad way, this also affect the reputation as well as cancellations. The growing trend of Hotels Industry is beneficial for Hotels but there are some problems too such as Rising Rate of Cancellation. The user cancels the booking of the hotel after seeing the reviews given by the people who booked the hotel already. Now a days, people expecting a better accommodation at the Hotel site, if people found any lag in accommodation then they give poor rating of that Hotel. So, if we looking at percentage of cancellation then we found that the percentage of cancellation is increasing day by day. From a survey Cancellation rate rose from under 33% in 2014 to 40% in 2018. Also, during the COVID-19 pandemic this percentage gets increased because peoples book their Hotels in very earlier time and after changing situation day by day during pandemic and State Government implementing Lockdown in particular State, people sudden changes their plans and cancel the Hotel Bookings.

The reviews and the feedbacks of the customer play an important role in the image as well as the revenue system of the hotel. But most of the travelers don't read all reviews. The system analyzes the reviews and feedback by the customers. The feedbacks of the customer are gathered from the hotel’s website and the stored as classes. As per the study, the model analyzes the overlooked information by the customers and takes some essential steps. Finally, after processing all the data collected an emotional analysis is done. The hotels thereby can take the required steps to improve their service.

The growth in IT industry also affects the Hotel Industry. However, this change is quite slow. Many researchers are focused on testing and applying new artificial intelligence technology and learning equipment in the hotel industry. The study offers a brief knowledge about the use of Machine Learning and its combined technology in the hotel and tourism industry. Machine learning is quite trending these days.

1.2. Problem Statement

The prediction of the hotel booking demand system is motivated to leverage the guest data and the booking behavior patterns to devise a strategy for hotel management, using Machine Learning. With the increasing day to day trend of hotel cancellation at the border time, it affects the local as well as international tourists more and more. Customers/Tourists do not have any idea about the pre-cancellation scenario of that particular hotel at the time of booking. Any user can know about the cancellation percentage of hotels in very early time. With the help of these predictions of hotel cancellations a user can take decisions about the booking of hotel/resort in lead time or not, also it will allow hotel managers/revenue managers to take action on bookings that are identified as "potentially going to be cancelled". Furthermore the development of these models can help contribute in the hotel revenue management.

1.3. Existing System

There is a central reservation system (CRS) used by hotels to manage their tasks and reservations. The Central Reservation System is a type of reservation software that is used to update and maintain information of a hotel pertaining to inventory and rates so that hotels are able to manage guest reservations and the process around such reservations in real time.

The reservation management software essentially automates the booking process, syncing up with the hotel’s website and social media so that guests can reserve easily and conveniently without having to visit another site. The reservation system can also link up to the channel manager, allowing hotels to distribute their availability to both online and agents in real time. These systems do not ensure promising reservations which in turn creates problems for tourists and affects the hotels’s business as well. The usual and biggest problem faced with the the existing system is that a hotel booking done by a customer/traveler can be cancelled anytime by the hotel without any prior notice. On the other hand any booking taken by the hotel can be cancelled by the customer without any prior notice which leads to a loss in the business since due to such bookings the hotels cannot provide the rooms to other visitors.

Based on studies, machine learning-based system prototypes are developed to solve the problem of cancellations that in turn lead to huge losses. Nowadays, an improvement has been observed in using ML and DL techniques. In opposition to the conventional methods, current methods are able to extract/select special features automatically from a raw dataset with higher performance results achieved.

1.4. Proposed System

In this proposed system the point is to utilize AI with the assistance of python to predict the probability of a hotel booking getting cancelled.

The approach for predicting the hotel booking demand involves combining multiple techniques to improve the accuracy and effectiveness of the model and identify the likelihood of bookings being cancelled which makes it possible for hotel managers to take measures to avoid these potential cancellations, such as offering services, discounts, or other perks. The prediction model enables the hotel managers to mitigate

revenue loss derived from booking cancellations and to mitigate the risks associated with overbooking (reallocation costs, cash or service compensations). The proposed system for the approach involves ensemble learning techniques. In this project the data is collected from Kaggle web site and all the data is collected from live dataset.
Here, Logistic Regression, Decision Tree, K Nearest Neighbors (KNN), Random Forest, AdaBoost Classifier models are applied and then check the performance of our algorithm.

Some of the advantages of the proposed system are:

1. A prediction algorithm can help improve the accuracy of hotel booking cancellation prediction by analyzing data on customer preferences and booking history.
2. A prediction algorithm can analyze large amounts of data quickly and efficiently in real-time.
3. A prediction algorithm can predict for the user about the cancellation percentage of hotels in very early time. With the help of these predictions of hotel cancellation user can take decision about the booking of hotel/resort in lead time or not.
4. A prediction algorithm allows hotel managers/revenue managers to take action on bookings that are identified as potentially going to be cancelled which in turn contributes to the hotel revenue management.

1.5. Aim and Purpose of the Project

Predicting the algorithm for a hotel booking and cancellation system requires evaluating the performance of each algorithm on a data set and the specific goals of the prediction algorithm. This may involve testing algorithms on subsets of data and measuring accuracy. It's also important to consider how the algorithm will scale as the data set grows over time. This project uses Logistic Regression, Decision tree and the

K Nearest Neighbors (KNN) algorithm, AdaBoost Classifier and Random Forest algorithms to predict the accuracy.

The purpose of a hotel booking demand and cancellation prediction is to use data and predictive modeling techniques to forecast the likelihood of a customer booking a hotel reservation or cancelling an existing reservation. This information can be used by hotels to make better decisions regarding inventory management, pricing, and staffing. By accurately predicting the probability of a customer booking a reservation, hotels can optimize their pricing strategies and allocate resources more efficiently. They can also identify high-demand periods and adjust their operations accordingly. On the other hand, by predicting the likelihood of a customer canceling a reservation, hotels can prepare for potential cancellations and minimize the impact on their operations and revenue. Overall, a hotel booking and cancellation prediction project can help hotels improve their revenue management, customer satisfaction, and operational efficiency.

1.6. Scope

The hotel industry is highly competitive, and hotels need to continuously optimize their operations and resources to stay ahead of the competition. The scope of the project aims at predicting future bookings and cancellations that can help hotels allocate their resources more efficiently, such as room inventory, staff, and amenities. This will help hotels to manage their operations more effectively, reducing costs and increasing efficiency. An accurate future bookings prediction can help hotels to optimize their pricing strategies, such as dynamic pricing or personalized offers, to maximize revenue.

Customer satisfaction helps hotels to prepare for potential disruptions and offer alternative solutions to customers, minimizing the negative impact on customer satisfaction. By analyzing historical data, hotels can identify patterns and trends in demand and use this information to forecast future demand more accurately, making it easier for hotels to plan ahead and improve their resource allocation. Overall, this project builds a reliable and accurate prediction model that can help hotels to make data-driven decisions and optimize their operations, leading to improved revenue, customer satisfaction, and competitive advantage.

1.7. Objectives

- Accurately predict booking and cancellation rates: The primary objective of the project is to develop a predictive model that can accurately forecast booking and cancellation rates for a given time period.

- Optimize pricing and revenue management: By accurately predicting booking rates, hotels can optimize their pricing strategies to maximize revenue and profitability.

- Efficient resource allocation: By predicting cancellations, hotels can manage their resources more efficiently, such as room inventory and staffing, minimizing waste and improving operational efficiency.

- Improved customer satisfaction: By predicting cancellations, hotels can offer alternative solutions to customers, minimizing the negative impact on customer satisfaction and loyalty.

- Better planning and decision-making: By analyzing historical data and forecasting future booking and cancellation rates, hotels can make more informed decisions regarding inventory management, pricing, and staffing, leading to better operational and financial outcomes. By optimizing operations and improving customer satisfaction, hotels can gain a competitive advantage in the market, leading to increased market share and profitability.

2. SYSTEM REQUIREMENT SPECIFICATIONS

2.1. Purpose

The purpose of system requirements is to define functional and non-functional specifications that a software system must meet in order to meet user needs. Predicting the hotel bookings and cancellations system is a project that will help to know which algorithm is best for customers and hotel owners to get personalized recommendations on hotel cancellation system and vice versa. The system uses machine learning algorithms to analyze customer data and suggest the cancellation scenario of hotels. System requirements help to establish a common understanding. It helps define the scope of the project by identifying the features that should be included in the system. Also helps to identify potential risks and issues that could impact the success of the project.

2.2. Feasibility Analysis

This research discusses the feasibility of applying machine learning approaches in hotel demand forecast. Two sets of empirical studies are designed: one with booking data from a single hotel with long booking histories, the other from multiple hotels with up to 14 days booking windows and other information including price, location, review score, etc. The results indicate that machine learning approaches outperform pick-up based models especially given long historical data.

Although lacking a systematic understanding of how machine learning approaches contribute to hotel demand forecast, the unique characteristics of some It is an important consideration as it involves assessing the technical aspects of the system, such as the algorithms used to analyze user data and generate recommendations. It involves evaluating the financial viability of the project, including the costs associated with development, implementation, and maintenance of the system, The system must comply with applicable laws and regulations related to data privacy. The data collected from users must be anonymized and stored securely to prevent any unauthorized access.

2.3. Hardware Requirements

• Processor	:	Intel core i3 or higher
• Ram	:	8GB or higher
• Hard Disk	:	80GB or higher

2.4. Software Requirements

• Operating System	:	Microsoft Windows 10 OS
• Language	:	Python
• Frameworks	:	Anaconda – Jupyter

2.5. Functional Requirements

- The system should be able to collect data about customer preferences, booking history, and other relevant information to provide personalized recommendations.
- The predicted algorithm should be able to process the booking history of hotels and resorts of tourists as well as owners and extract relevant attributes.
- The predicted algorithm should be able to analyze user data and provide cancellation predictions based on their past bookings.
- The system should be able to personalize cancellation probability based on individual user booking history.
- Helps improve the onsite experience by making probable cancellation predictions for different categories of people.

- Predicted algorithm improves hotel booking experience which helps the users to have a comforting and promising experience and a profitable business for the hotel owners.

2.6. Non-Functional Requirements

- **Performance:** The system should be able to process large volumes of data quickly and provide predictions in real-time, as hotel staff need to make decisions quickly to optimize inventory management and pricing.

- **Accuracy:** The system should provide accurate predictions to ensure that hotel staff can rely on them to make informed decisions. A high level of accuracy can be achieved through appropriate data pre-processing, feature engineering, and model development.

- **Scalability:** The system should be scalable, able to handle increased demand as the hotel grows and expands its operations. This may involve using cloud-based infrastructure or distributed systems to ensure the system can handle increased traffic and data volumes.

- **Availability:** The system should be available 24/7, as hotel staff may need to access the system at any time to make decisions about inventory management and pricing.

- **Reliability:** The system should be reliable, ensuring that it does not crash or fail unexpectedly, which could result in lost bookings and revenue for the hotel.

- **Usability:** The system should be easy to use and understand, with a user-friendly interface that requires minimal training for hotel staff to use effectively.

- **Security:** The system should be secure, ensuring that the collected data and the models developed from it are protected from unauthorized access or misuse. This may involve implementing appropriate encryption, access control, and data anonymization measures.

- **Maintainability:** The system should be easy to maintain, with clear documentation and well-organized code that allows for easy debugging and updates as necessary.

- **Compatibility:** The system should be compatible with the hotel's existing IT infrastructure, software, and hardware, to ensure seamless integration with other systems and tools used by hotel staff.

- **Compliance:** The system should comply with any relevant industry regulations, such as data protection laws, and ethical considerations, to ensure the project is conducted responsibly and ethically.

3. SOFTWARE

3.1 About the Software

Python is a high-level, interpreted programming language that was first released in 1991. It was designed to be easy to read, write, and understand, with a clear and concise syntax. Python is known for its versatility, and it can be used for a wide range of applications, including web development, data science, machine learning, automation, and more.

Python was developed in the late 1980s and early 1990s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages. Python's source code is available under the GNU General Public License (GPL) and is now maintained by a core development team at the institute, with Guido van Rossum still holding a vital role in directing its progress.

Python's popularity has been growing rapidly in recent years. According to the TIOBE index, Python is currently the third most popular programming language, behind only Java and C. Python's popularity can be attributed to its ease of use, versatility, and large community. Its simplicity and readability make it a popular choice for beginners, while its powerful features make it a favorite among experienced developers. Python's popularity has also been driven by its use in fields such as data science, machine learning, and artificial intelligence, where it has become a standard tool. As a result, Python is now used by a wide range of companies and organizations, from small startups to large tech giants like Google, Amazon, and Facebook.

Some of the key features of Python include:

- Simple and clear syntax: Python's syntax is designed to be easy to read and understand which makes it a popular choice for beginners.
- Dynamic typing: Python is dynamically typed, which means that the type of a variable is determined at runtime.
- Object-oriented programming: Python supports object-oriented programming, which allows developers to create reusable code and encapsulate functionality in objects.
- Large standard library: Python comes with a large standard library that includes modules for tasks such as web development, data processing, networking, and more.
- Third-party libraries: Python has a large active community that has created many third-party libraries for tasks including data science, machine learning, and more.

- Cross-platform: Python is cross-platform, which means that code written on one platform can be run on another platform without modification.
- Interpreted language: Python is an interpreted language, which means that code can be executed directly without the need for a compilation step.
- High-level language: Python is a high-level language, which means that it provides abstractions that make it easier to write complex code. For example, Python's built-in data structures such as lists and dictionaries make it easy to work with collections of data.
- Rapid prototyping: Python is a popular choice for rapid prototyping, as it allows developers to quickly write and test code. This makes it a great choice for projects that require a quick turnaround time.
- Community support: Python has a large and active community, which means that developers can easily find help, support, and resources online. The community also contributes to the development of new libraries and tools, which can be used to extend Python's functionality.
- Compatibility: Python is compatible with a wide range of other technologies and platforms.

Python can run on a wide variety of platforms, including Linux and Mac OS X. The most up-to-date and current source code, binaries, documentation, news, etc., are available on the official website of Python at https://www.python.org. To install Python on a Windows machine, you can follow the steps on the website to download and run the installer.

Python has similarities to Perl, C, and Java, but also has some definite differences between these languages. Python allows for both interactive mode programming and script mode programming. In interactive mode, you can execute Python code directly in the interpreter, while in script mode, you write code in a .py file and execute it using the Python interpreter. Overall, Python is a versatile and powerful programming language that is great for both beginners and experienced programmers alike.

Use of Python

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Advantages of Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

More about Python

The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Importing Libraries:

Numpy - A basic package for scientific computing in Python. It supports multidimensional array objects, various derived objects (such as masked arrays and matrices), as well as mathematics, logic, shape manipulation, sorting, selection, I/O, A Python library that provides a set of routines for quick operations on arrays, such as discretes. Fourier transforms, basic linear algebra, basic statistical operations, random simulations, and more.
importing - import numpy as np

Pandas - The most commonly used open-source Python package for data science/data analysis and machine learning tasks. It is based on another package called Numpy that provides support for multidimensional arrays. One of the most popular data wrangling packages, Pandas works well with many other data science modules in the Python ecosystem and is typically available on everything from those that come with operating systems to the more commercial ones. Included with the Python distribution are providers such as Active State and Active Python. Pandas is primarily used for data analysis and related manipulation of tabular data in data frames.
importing - import pandas as pd

Matplotlib - A cross-platform data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it provides a viable open-source alternative to MATLAB. Developers can also embed charts in GUI applications using matplotlib's API (application programming interface). Most Python matplotlib scripts are structured so that you can create visual representations of your data with just a few lines of code. The Matplotlib script layer overlays two APIs:

- The Pyplot API is a hierarchy of Python code objects with matplotlib.pyplot as the apex.
- His OO (Object Oriented) API collection of objects that can be assembled more flexibly than pyplot. This API provides direct access to Matplotlib's backend layer.

importing - import matplotlib.pyplot as plt

Seaborn - Library for creating statistical graphs in Python. Built on Matplotlib and tightly integrated with pandas data structures. Seaborn helps you explore and understand your data. Its presentation capabilities manipulate data frames and arrays that contain entire data sets and perform the necessary semantic mapping and statistical aggregation internally to create meaningful presentations. A record-oriented, declarative API allows you to focus on the meaning of the different elements of the chart rather than the details of how to draw it.

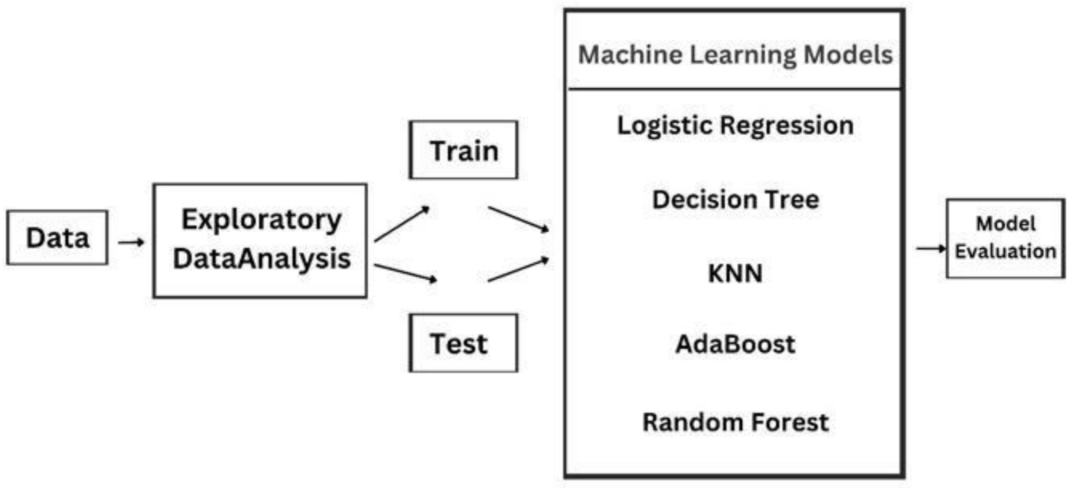
Importing - import seaborn as sns

4. SYSTEM ANALYSIS AND REQUIREMENTS DOCUMENTATION

4.1 Overview

System analysis and requirements for predicting hotel bookings and cancellations involves the process of identifying, modelling, and defining system requirements and specifications. The first step in system analysis is gathering and documenting system requirements. This includes identifying and understanding user needs and documenting the functional and non-functional requirements of the system. Once knowing the requirements, the next step is to create a model of system. This includes creating diagrams, flowcharts, and other visual representations of the system. Then create a simplified version of the system that can use to test and refine the requirements and design. The fourth step is to test the system to ensure that it meets requirements and specifications. In the system analysis process, it is important to consider various factors such as system scalability, maintainability, and reliability. It's also important to ensure that system meets various industry standards and regulations, such as privacy and security regulations.

4.2. Proposed System Architecture



4.3. Module Description

- Feature Selection
- Model Selection, Train & Test
- Model Accuracy
- Model Comparison

4.3.1. Module – I - Feature Selection

The first step is to select the features about user bookings, behavior and other relevant information which is useful to generate predictions. Feature selection is the process of selecting the most relevant features for predicting hotel bookings and cancellations. Some features include room type, price, location, and availability. The goal of feature selection is to identify the features that have the greatest impact on the prediction task and to eliminate irrelevant or redundant features.

Methodology:

Input: Collected data is given to the system
Output: The given data is used to select the necessary features
Process:
Feature selection is a process of selecting the most relevant and informative features from a dataset to improve the performance of machine learning models. In the context of a hotel booking and cancellation prediction system, feature selection is important because not all features in the dataset may be relevant for predicting hotel bookings and cancellations.

The process of feature selection typically involves two main steps: feature ranking and feature subset selection.

Feature ranking involves assigning a score or importance value to each feature in the dataset based on its relevance to the prediction task. This can be done using various statistical or machine learning techniques, such as correlation analysis, mutual information, or tree-based feature importance. Features with higher importance scores are considered more relevant and informative for the prediction task.

Feature subset selection involves selecting a subset of the most important features from the dataset based on their importance scores. This can be done using various techniques, such as greedy algorithms, recursive feature elimination, or genetic algorithms. The goal is to select a subset of features that maximizes the prediction performance of the machine learning model while minimizing the computational resources required for training and inference.

There are several benefits to performing feature selection in a hotel booking and cancellation prediction system. First, it can help improve the accuracy and robustness of the machine learning model by reducing the amount of noise and irrelevant information in the dataset. Second, it can help reduce the computational resources required for training and inference by reducing the dimensionality of the dataset. Third, it can help improve the interpretability and explainability of the machine learning model by focusing on the most relevant and informative features. Overall, feature selection is an important step in the machine learning pipeline for any prediction system and can greatly improve the performance and interpretability of the model.

4.3.2. Module – II - Model Selection, Train & Test

Model selection involves choosing the appropriate machine learning algorithm for the prediction task. There are many algorithms to choose from, including logistic regression, decision trees, KNN, AdaBoost and Random Forest. The choice of algorithm depends on the nature of the data and the prediction task, as well as other factors such as interpretability and scalability.

Methodology:

Input: Selected features data is given to the system

Output: Machine learning models are selected, trained and tested

Process:

Model selection is a critical step in the development of a machine learning-based hotel booking and cancellation prediction system. Model selection involves choosing the best machine learning algorithm or model that can accurately predict hotel bookings and cancellations.

There are several machine learning algorithms or models that can be used for the hotel booking and cancellation prediction system, such as logistic regression, decision trees, random forests, support vector machines, and neural networks. Each of these models has its own strengths and weaknesses, and their performance may vary depending on the characteristics of the dataset and the prediction task.

Model selection typically involves comparing the performance of multiple machine learning models on the same dataset using a performance metric, such as accuracy, precision, recall, F1 score, or area under the receiver operating characteristic curve (AUC-ROC). The performance of each model can be evaluated using techniques such as cross-validation, which involves splitting the dataset into training and validation sets and testing the model on multiple splits of the data.

Model selection can also involve hyperparameter tuning, which involves selecting the best hyperparameters for each machine learning model. Hyperparameters are parameters that are set before the training of the model, such as learning rate, regularization strength, or tree depth, and can have a significant impact on the performance of the model.

There are several factors to consider when selecting a machine learning model for a hotel booking and cancellation prediction system, such as the interpretability of the model, the computational resources required for training and inference, and the performance on different subsets of the data. The selected model should be able to accurately predict hotel bookings and cancellations while also being interpretable and scalable to large datasets.

Once the model has been selected, it needs to be trained using the preprocessed data. During training, the model learns to predict hotel bookings and cancellations based on the selected features. This can involve techniques such as gradient descent, backpropagation, or tree pruning. The model's performance is evaluated using appropriate metrics such as accuracy, precision, recall, and F1 score. Regularization techniques may also be used to prevent overfitting and improve the model's generalization performance.

After training, the model needs to be validated and tested to ensure its generalization performance. Validation involves using cross-validation techniques to evaluate the model's performance on unseen data, while testing involves evaluating the model on a holdout dataset. The performance of the model is evaluated using various metrics such as accuracy, precision, recall, and F1 score. Model selection is also performed at this stage to select the best-performing model.

4.3.3. Module – III – Model Accuracy

The accuracy of the model is fundamentally the total number of correct predictions divided by a total number of predictions. The precision of a class defines how trustable is the outcome is when the model answers that a point belongs to that class.

We apply few metrics to evaluate the performance of each model like confusion matrix, accuracy score, F-score, etc. The first metric is the confusion matrix. A simple and yet good metric that always helps when dealing with classification problems is the confusion matrix. A confusion matrix is a table that is used to represent the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. Thus, it is a terrific starting point for any classification model evaluation. The accuracy score measures the fraction of the number of correct predictions of total predicted samples. The value of Accuracy is between 0 and 1. The greater the accuracy score is, the better the model performs. The F1 -score is the harmonic mean of precision and recall. Precision measures how many samples of a predicted class are correctly predicted. Recall measures how many samples of an actual class are correctly predicted. F1 of a high quality model is close to 1.

A decision tree algorithm can be used in a hotel demand prediction system by using a set of rules to determine the best recommendations for a given user. A hotel demand prediction system using k-nearest neighbors (KNN) is a type of machine learning algorithm that is used to predict booking cancellations for users based on their previous interactions, bookings and cancellations. AdaBoost Classification is used in a hotel demand prediction systems to predict cancellation probability based on their historical bookings and other features. Random Forest on hotel demand prediction systems have been shown to be effective in various domains and also require a significant amount of training data and feature engineering to achieve good performance.

Methodology:

Input: Pre-processed data is given to the system

Output: Models are trained and accuracy is known

Process:

After pre-processing and splitting the dataset into training and testing sets. The training set is used to build the model algorithms, while the testing set is used to evaluate its performance. Extract and select relevant features that will be used to train the model. Choose a suitable machine learning model that fits the problem such as KNN (K Nearest Neighbors), Decision Tree, AdaBoost Classifier, Logistic Regression and Random Forest. Each algorithmic model is generated and accuracy is calculated.

The nearest neighbor algorithm is one of the most straightforward non-parametric decision rules. Nearest neighbor can be used in the classification problem, which assigns an unclassified observation into the category to the nearest sample. It can also be applied in predicting the test value by taking the average of the k closest neighbors. Given the focus of this current research (the hotel demand, a continuous numeric variable), this section will target the regression perspective. To calculate the distance between the targeting test value x and existing training observation yi in a p dimension space, the simplest instance is calculating their Euclidean distance.

Tree-based models partition the feature space into a group of rectangles and fit a simple model in each space. A decision tree is composed of multiple judgment nodes, representing a mapping relationship between the attributes and values. To build a decision tree, we start by finding the optimal split for attributes which generates the largest information gain. Currently, the primary measure metrics for information gain are entropy and the Gini impurity. Decision tree models have many attractive features. A major advantage of the decision tree is its interpretability. Conducting forecasts using decision tree models closely mirrors human-being’s decision-making process. This benefit is particularly useful in hotel management practice since it is easy to understand by the general audience without a strong statistical background. Another advantage of the decision tree is it can efficiently deal with qualitative predictors without creating dummy variables, which is superior to K-NN and neural network models.

Logistic regression comes under the most popular Supervised Machine Learning algorithms. Logistic regression predicts the categorical dependent variable using a given set of independent variables. The categorical dependent variable should be either Yes or No, 0 or 1, true or false, etc. Logistic regression is much comparable to Linear Regression only implementation is different. Linear Regression solves the Regression problems, and Logistic regression is used for solving the classification problems. Instead of fitting the regression line, In logistic regression, we fit the sigmoid S function, which predicts two maximum values (0,1).

Random Forest is a popular supervised machine learning algorithm. The random forest algorithm implements both Classification and Regression problems in ML. The Random Forest is a classifier that includes several decision trees instead of relying on one decision tree, the random forest takes the prediction from each tree, and based on the majority votes of predictions, it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

4.3.4. Module – IV – Model Comparison

The above all the information is considered and the prediction system uses the right algorithm to predict the cancellation of a booking. When a user makes a hotel booking, it is said that the prediction system's algorithm works accurately based on the user's past bookings.

Methodology

Input: Comparing different models accuracy.
Output: Highest accuracy is chosen
Process

Choose a hybrid approach that suits for the problem.

The combination of accuracies obtained from various algorithms. The ensemble technique with the highest accuracy is selected
The chosen approach needs to be further refined to achieve the best possible performance.

4.4. UML (Unified Modelling Language) Diagrams:

UML stands for Unified Modelling Language and is a standardized visual language.

Presentation and design of software systems. UML diagrams provide a visual representation. Various aspects of software systems such as structure, behavior and interaction.

4.4.1 Use Case Diagram

A use case diagram is a graphical representation of the functionalities or actions that a system can perform. It can create a broad, high-level view of the relationship between use cases, actors involved, and systems being performed. From the use case diagram below, it can be seen that use cases are represented by oval shapes and lines that show at which point an actor/user participates and interacts with their corresponding use case. It can also be analyzed how each actor is involved within the entire process and where they're excluded. In the context of a hotel demand prediction system, a use case diagram can illustrate the interactions between the system and its users, and the actions that the system can perform to provide predictions on cancellations with respect to booking history. This use case involves the user side as well as the hotel side. The diagram shows how a booking and cancellation can be made by the user/tourist as well as by the hotel receptionist. Other services like viewing a reservation, editing the reservation, recording a payment, etc. can be done from both ends in the online hotel booking and cancellation system since both the customer and owner have open options or control over the process.

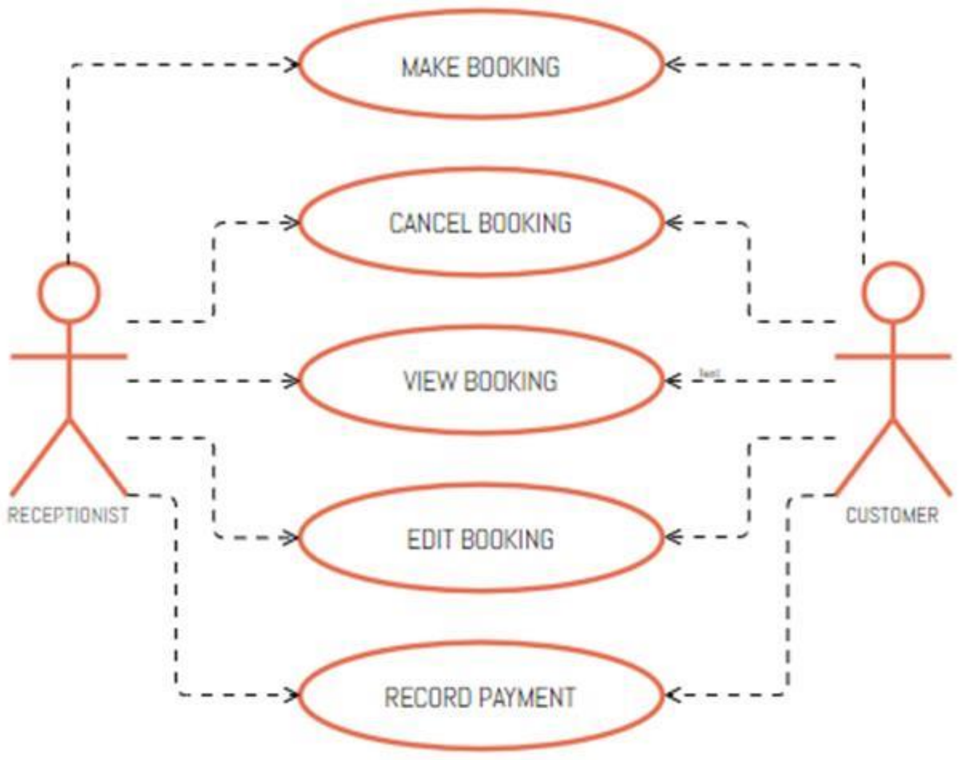


Fig 1: USE CASE DIAGRAM OF HOTEL MANAGEMENT

4.4.2. Activity Diagram:

When predicting algorithms for hotel demand prediction system, activity diagrams can be used to model flow system processes.

1. User enters collected data
2. The system collects data about users' past bookings, booking and cancellation history and other related information.
3. The system applies machine learning algorithms to analyze user data and preferences.
4. The system compares the model accuracy.
5. Finally, a suitable algorithm is selected.

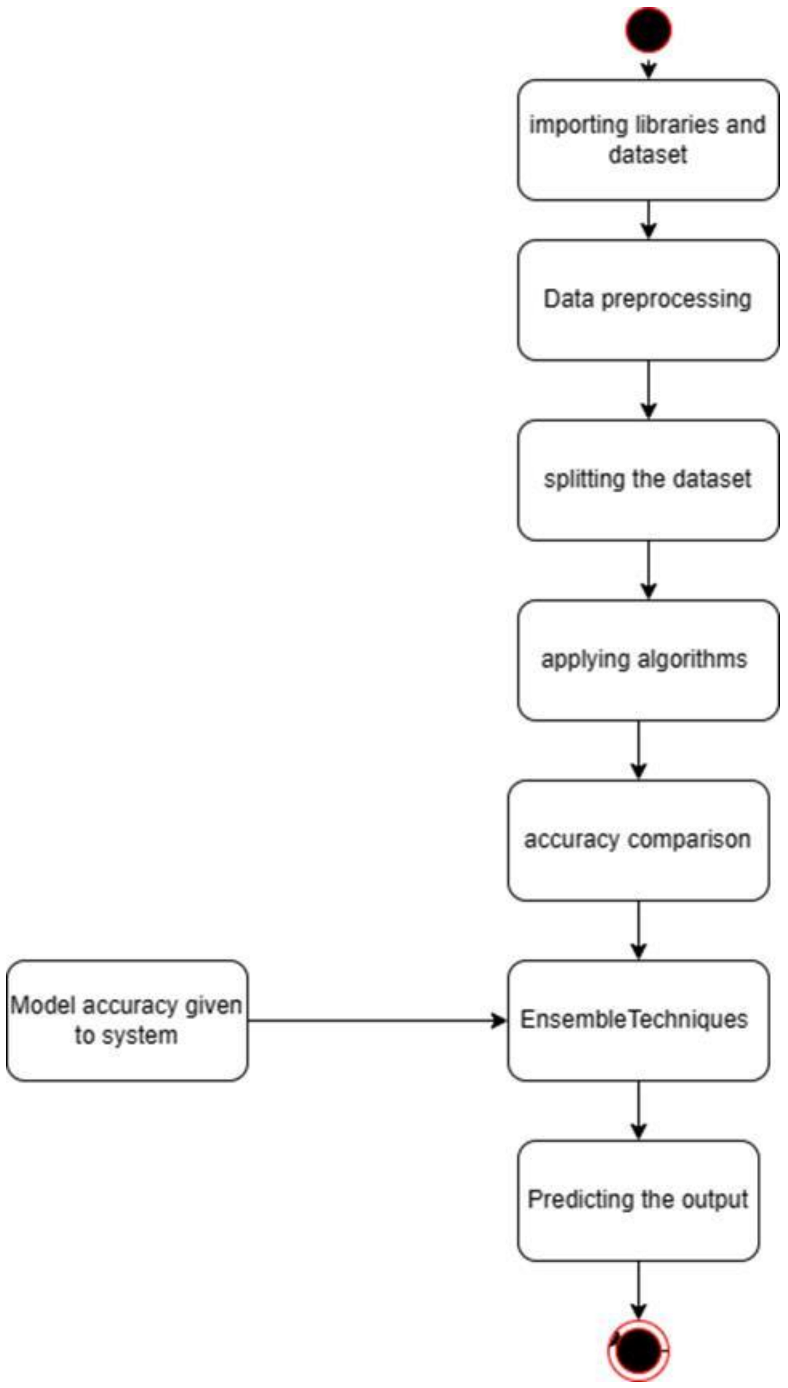


Fig 2: ACTIVITY DIAGRAM FOR HOTEL DEMAND PREDICTION

4.4.3 Sequence Diagram

A hotel demand prediction system can use sequence diagrams to represent interactions between users and the system. Sequence diagrams help hotels and the customers to understand the sequence of bookings and weather they are promising or not. Modeling the system in this way can help predict suitable algorithms. Additionally, sequence diagrams can be used to communicate system behavior to stakeholders who may not have a technical background.

Sequence diagrams in UML are used to illustrate the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines and the messages that they exchange overtime during the interaction.

The user can view the hotel lists, view the price of a selected hotel, book the hotel, and make payments for the booking. Users can book hotels or rooms after filling out some simple forms. Users can manage all the booked history as well.

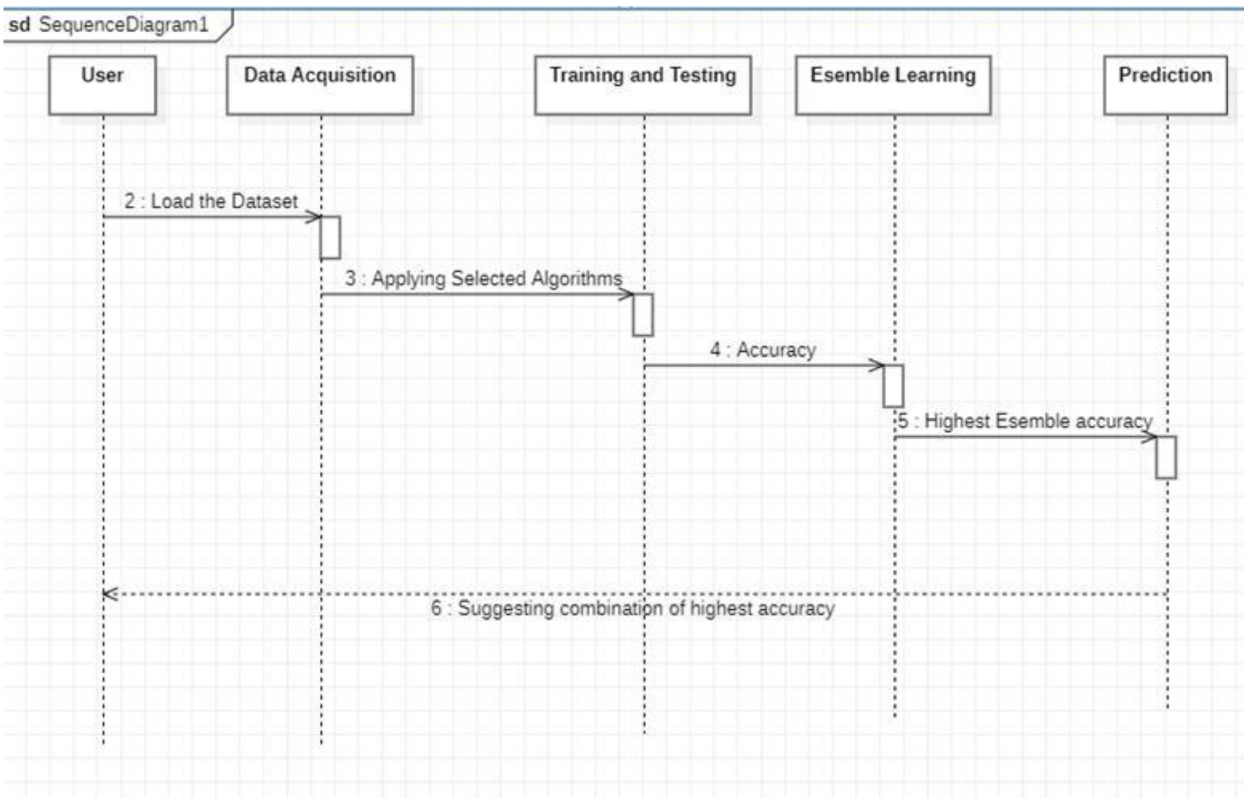
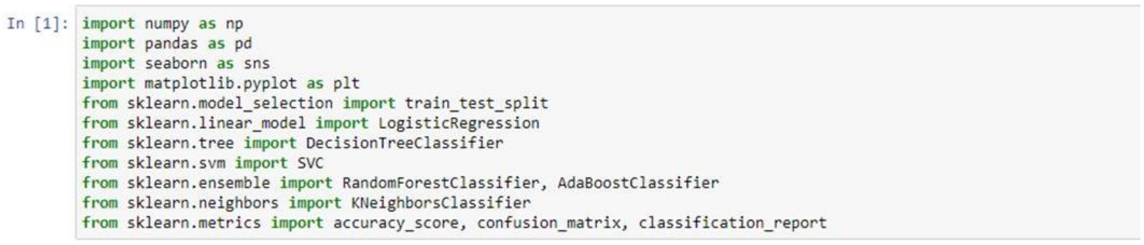


Fig 3: SEQUENCE DIAGRAM FOR HOTEL MANAGEMENT SYSTEM

5. SYSTEM DESIGN AND DOCUMENTATION

Importing Libraries:

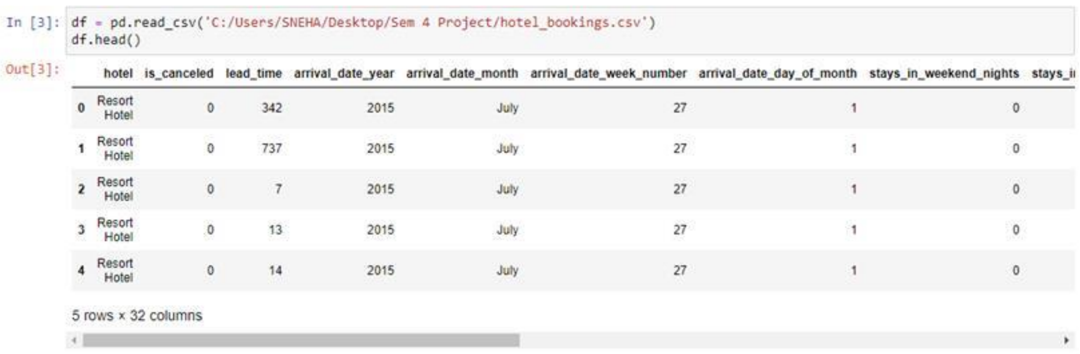
When working with machine learning, it's important to import the necessary libraries so that can efficiently manipulate the data and build machine learning models. Here are some of the main libraries used in machine learning models:



```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.svm import SVC
        from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Data Collection:

Data collection is a key step in predicting. The system relies on data about user bookings, hotel features, customer features and other relevant information to provide users with prediction recommendations. Data collection refers to the process of collecting and preparing data used to train and test machine learning models. The quality and quantity of collected data play an important role in determining the accuracy and reliability of machine learning models. The dataset is collected from Kaggle website.



```
In [3]: df = pd.read_csv('C:/Users/SNEHA/Desktop/Sem 4 Project/hotel_bookings.csv')
        df.head()
```

Out[3]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_i
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	

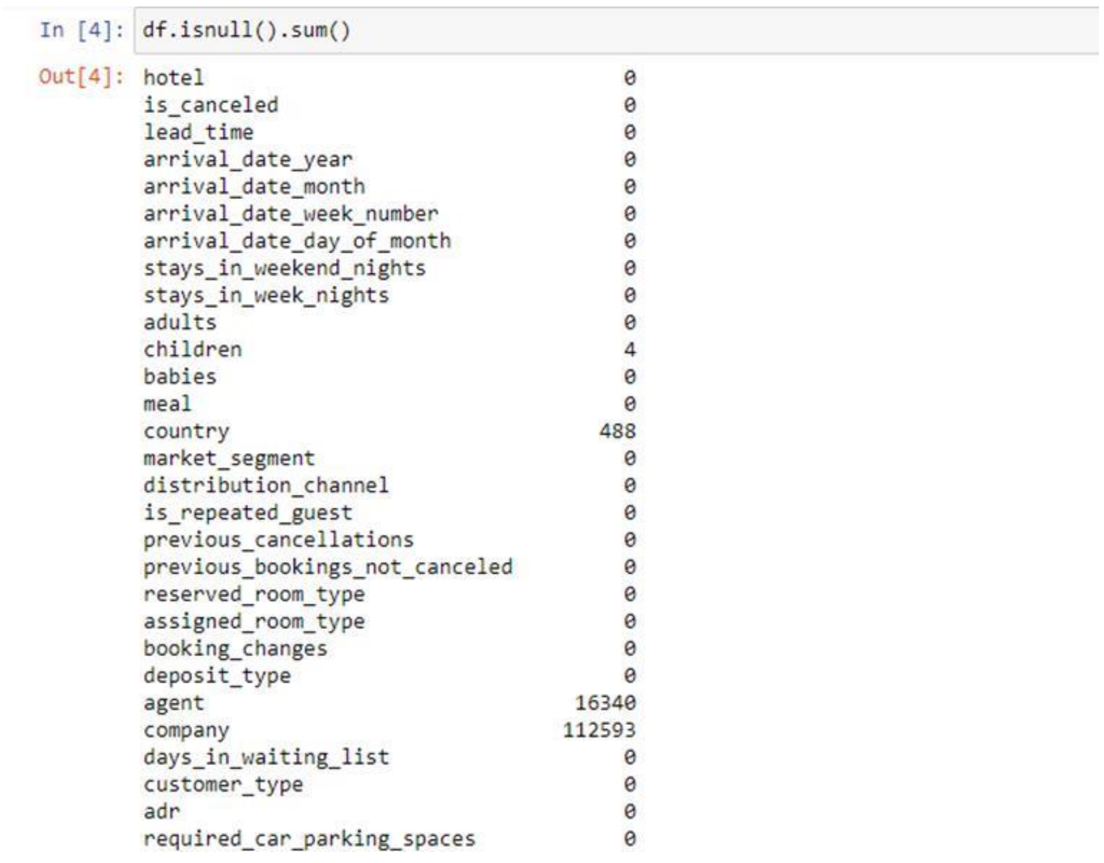
5 rows × 32 columns

Data Pre-processing:

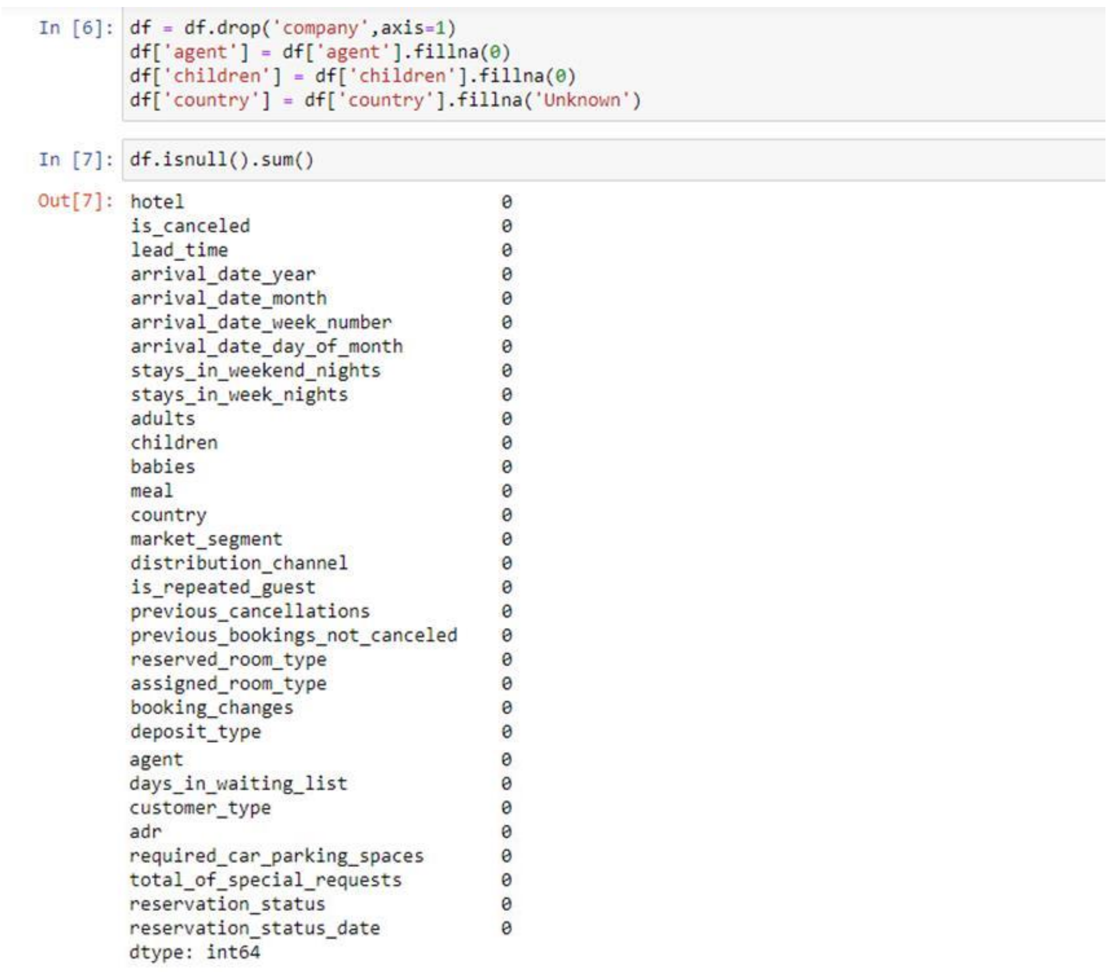
Data pre-processing is an important step in preparing data for machine learning algorithms. This includes cleaning, transforming, and organizing data to make it easier for machine learning models to learn from the data.

Handling Missing Data: Missing data is a common problem in many datasets. There are several ways to handle missing data, such as removing missing values, padding missing values with default values, and using imputation methods to estimate missing values.

Handling Outliers: Outliers are data points that differ significantly from other data points in the dataset. Outliers can be problematic for some machine learning algorithms, so it's important to handle them properly. This includes removing outliers, transforming data to reduce the impact of outliers, or using robust statistical methods that are less susceptible to outliers.



```
In [4]: df.isnull().sum()

Out[4]: hotel                                  0
        is_canceled                            0
        lead_time                              0
        arrival_date_year                      0
        arrival_date_month                     0
        arrival_date_week_number               0
        arrival_date_day_of_month              0
        stays_in_weekend_nights                0
        stays_in_week_nights                   0
        adults                                 0
        children                               4
        babies                                 0
        meal                                   0
        country                              488
        market_segment                         0
        distribution_channel                   0
        is_repeated_guest                      0
        previous_cancellations                 0
        previous_bookings_not_canceled         0
        reserved_room_type                     0
        assigned_room_type                     0
        booking_changes                        0
        deposit_type                           0
        agent                              16340
        company                           112593
        days_in_waiting_list                   0
        customer_type                          0
        adr                                    0
        required_car_parking_spaces            0
```



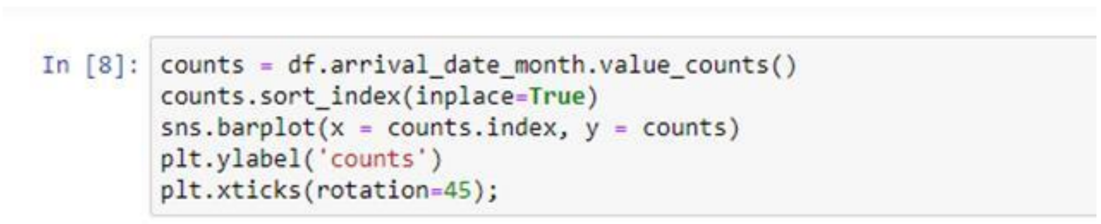
```
In [6]: df = df.drop('company',axis=1)
        df['agent'] = df['agent'].fillna(0)
        df['children'] = df['children'].fillna(0)
        df['country'] = df['country'].fillna('Unknown')

In [7]: df.isnull().sum()

Out[7]: hotel                             0
        is_canceled                       0
        lead_time                         0
        arrival_date_year                 0
        arrival_date_month                0
        arrival_date_week_number          0
        arrival_date_day_of_month         0
        stays_in_weekend_nights           0
        stays_in_week_nights              0
        adults                            0
        children                          0
        babies                            0
        meal                              0
        country                           0
        market_segment                    0
        distribution_channel              0
        is_repeated_guest                 0
        previous_cancellations            0
        previous_bookings_not_canceled    0
        reserved_room_type                0
        assigned_room_type                0
        booking_changes                   0
        deposit_type                      0
        agent                             0
        days_in_waiting_list              0
        customer_type                     0
        adr                               0
        required_car_parking_spaces       0
        total_of_special_requests         0
        reservation_status                0
        reservation_status_date           0
        dtype: int64
```

Data Visualization:

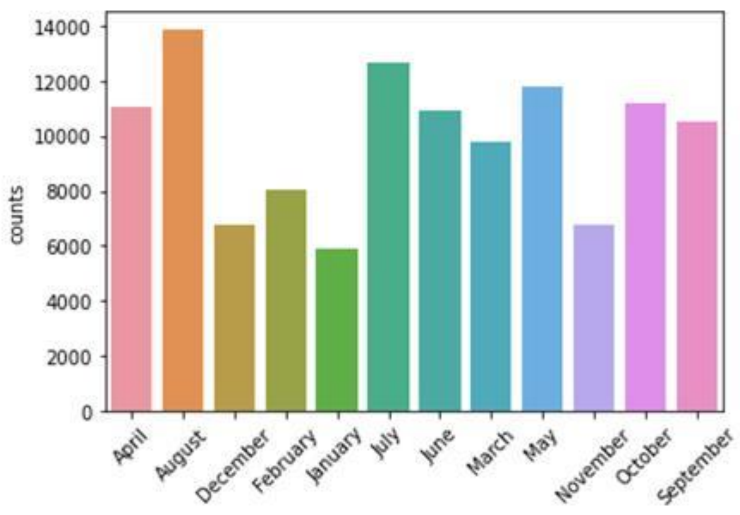
When working with data, it can be difficult to really understand the data if it is only in tabular form. To understand exactly what the data is telling us, to better organize the data and choose the right model, we need to visualize or image the data.

The process of finding trends and relationships in data through a visual representation is called data visualization. To do data visualization in Python, can use various Python data visualization modules such as Matplotlib, Seaborn, and Plotly. Data visualization is the area of data analysis that deals with visual representation of data. It is an effective way to present data graphically and communicate the conclusions drawn from the data.

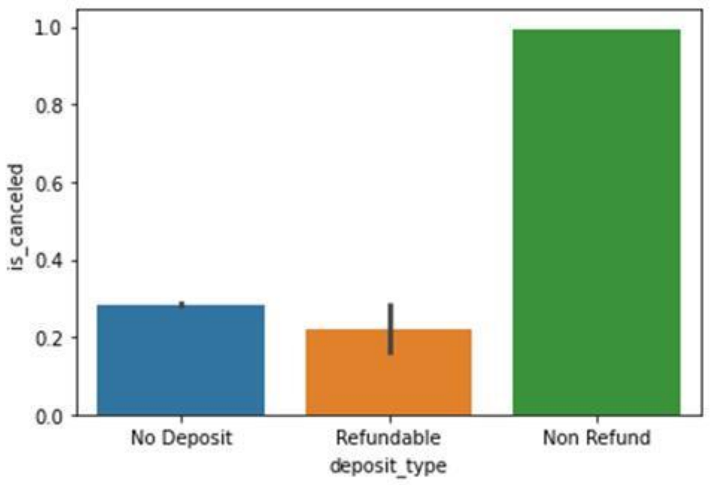
Data visualization allows you to get a visual summary of your data. Images, maps and graphs make it easier for the human mind to process and understand given data. Data visualization plays an important role in representing both small and large datasets. It's especially useful when you have an impossibly large dataset. Python offers several plotting libraries, such as Matplotlib, Seaborn, and many other data visualization packages, to help you create informative, customized, and attractive graphs to visualize your data in the easiest and most effective way. It has various functions for displaying the data graphically.



```
In [8]: counts = df.arrival_date_month.value_counts()
        counts.sort_index(inplace=True)
        sns.barplot(x = counts.index, y = counts)
        plt.ylabel('counts')
        plt.xticks(rotation=45);
```



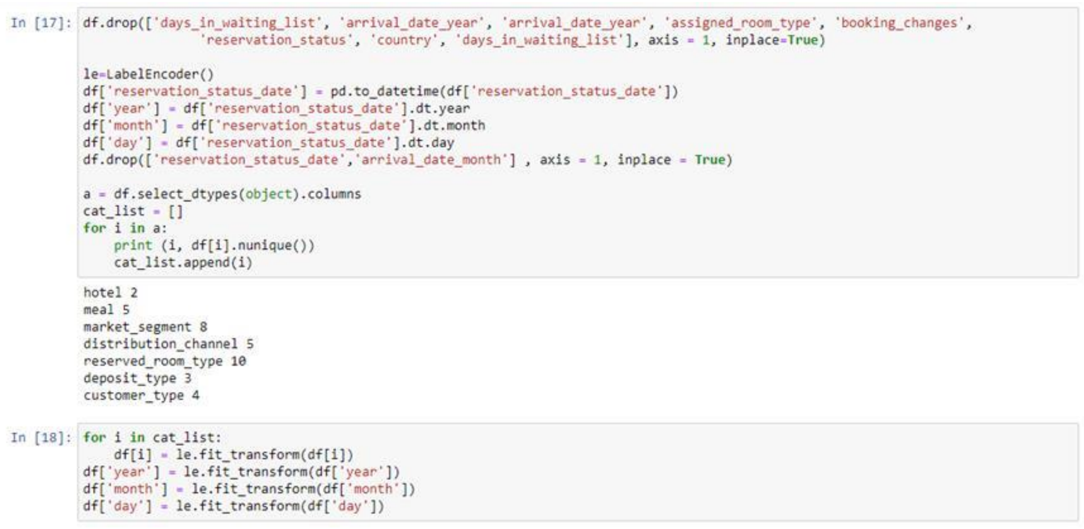
```
In [14]: sns.barplot(y="is_canceled", x="deposit_type", data=df)

Out[14]: <AxesSubplot:xlabel='deposit_type', ylabel='is_canceled'>
```



Applying LabelEncoder

In machine learning, we usually deal with datasets that contain multiple labels in one or more than one columns. These labels can be in the form of words or numbers. To make the data understandable or in human-readable form, the training data is often labelled in words. Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

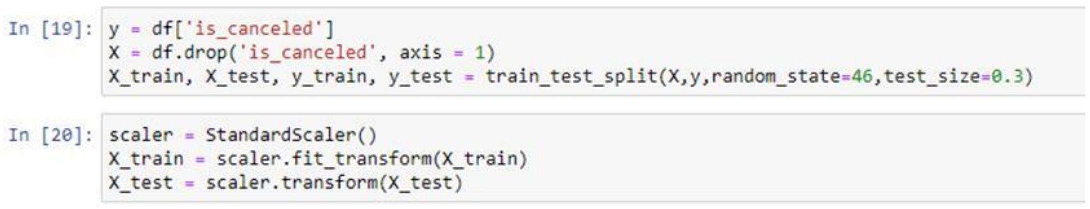
The approach is very simple and it involves converting each value in a column to a number. Consider a dataset of bridges having a column names bridge-types having below values. Though there will be many more columns in the dataset, to understand label-encoding, we will focus on one categorical column only.



```
In [17]: df.drop(['days_in_waiting_list', 'arrival_date_year', 'arrival_date_year', 'assigned_room_type', 'booking_changes',
                        'reservation_status', 'country', 'days_in_waiting_list'], axis = 1, inplace=True)

         le=LabelEncoder()
         df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date'])
         df['year'] = df['reservation_status_date'].dt.year
         df['month'] = df['reservation_status_date'].dt.month
         df['day'] = df['reservation_status_date'].dt.day
         df.drop(['reservation_status_date','arrival_date_month'] , axis = 1, inplace = True)

         a = df.select_dtypes(object).columns
         cat_list = []
         for i in a:
             print (i, df[i].nunique())
             cat_list.append(i)

         hotel 2
         meal 5
         market_segment 8
         distribution_channel 5
         reserved_room_type 10
         deposit_type 3
         customer_type 4

In [18]: for i in cat_list:
             df[i] = le.fit_transform(df[i])
         df['year'] = le.fit_transform(df['year'])
         df['month'] = le.fit_transform(df['month'])
         df['day'] = le.fit_transform(df['day'])
```

Splitting Data:

A training set is used to train a machine learning model. During the training process, the model learns patterns and relationships in the data and uses that information to make predictions about new, unseen data. The goal of the training process is to optimize the parameters of the model so that it can accurately predict the target variable on new data. After the model is trained, it is evaluated on the test set. A test set is a subset of data that was not used during training and contains known values for target variables. The model makes predictions on the test set, and model performance is evaluated by comparing the predicted values to the actual values on the test set. The main advantage of data splitting is that it allows us to estimate how well the model will perform on new unseen data. By evaluating the model on a different test set, you can get an idea of how well the model generalizes to new data and avoid overfitting.

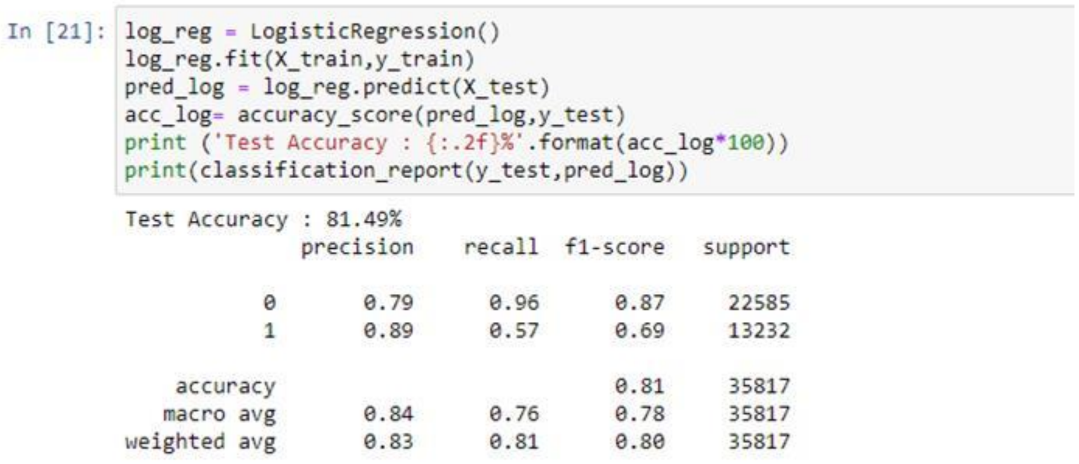


```
In [19]: y = df['is_canceled']
         X = df.drop('is_canceled', axis = 1)
         X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=46,test_size=0.3)

In [20]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

Training the algorithms:

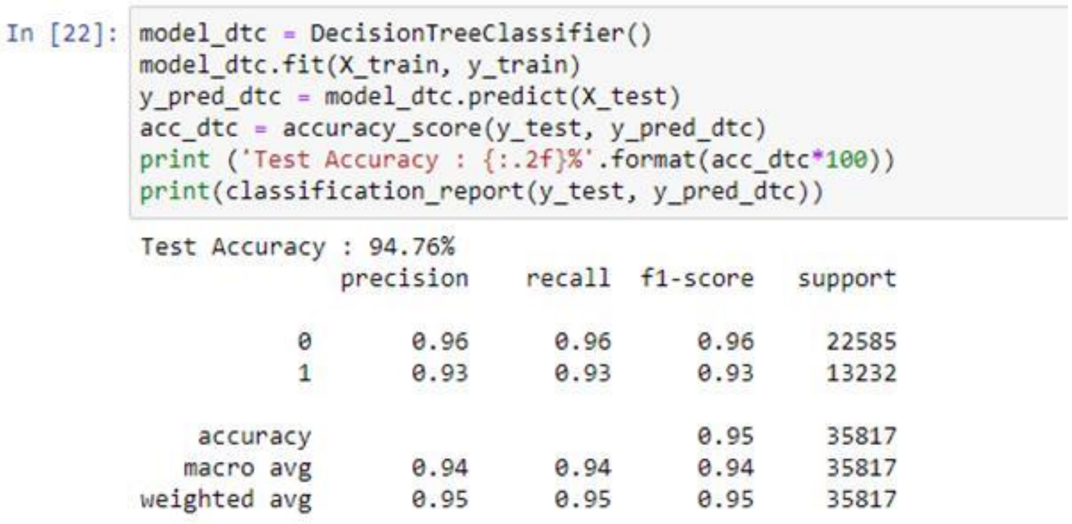
The selected algorithm fits the model using the training and test datasets and the system gives training and test scores.

Logistic Regression:



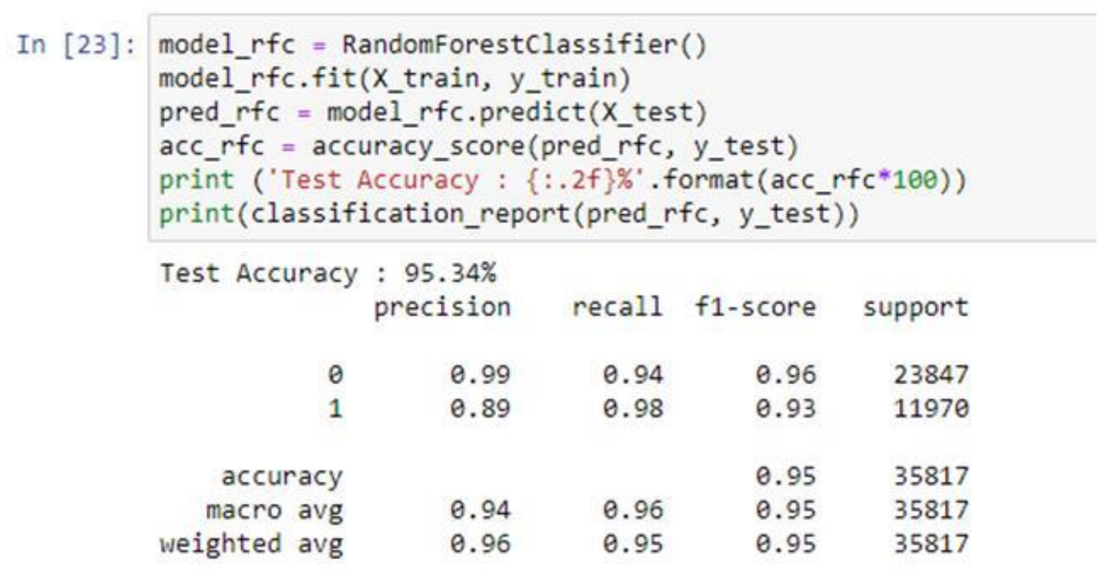
```
In [21]: log_reg = LogisticRegression()
         log_reg.fit(X_train,y_train)
         pred_log = log_reg.predict(X_test)
         acc_log= accuracy_score(pred_log,y_test)
         print ('Test Accuracy : {:.2f}%'.format(acc_log*100))
         print(classification_report(y_test,pred_log))

         Test Accuracy : 81.49%
                       precision    recall  f1-score   support

                    0       0.79      0.96      0.87     22585
                    1       0.89      0.57      0.69     13232

             accuracy                           0.81     35817
            macro avg       0.84      0.76      0.78     35817
         weighted avg       0.83      0.81      0.80     35817
```

Decision Tree Algorithm:



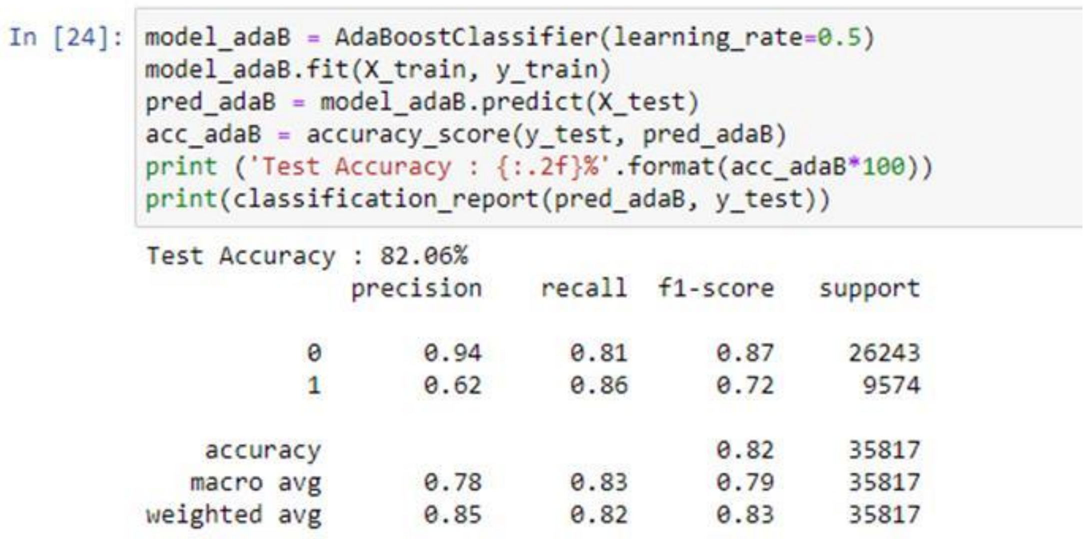
```
In [22]: model_dtc = DecisionTreeClassifier()
         model_dtc.fit(X_train, y_train)
         y_pred_dtc = model_dtc.predict(X_test)
         acc_dtc = accuracy_score(y_test, y_pred_dtc)
         print ('Test Accuracy : {:.2f}%'.format(acc_dtc*100))
         print(classification_report(y_test, y_pred_dtc))

         Test Accuracy : 94.76%
                       precision    recall  f1-score   support

                    0       0.96      0.96      0.96     22585
                    1       0.93      0.93      0.93     13232

             accuracy                           0.95     35817
            macro avg       0.94      0.94      0.94     35817
         weighted avg       0.95      0.95      0.95     35817
```

Random Forest:



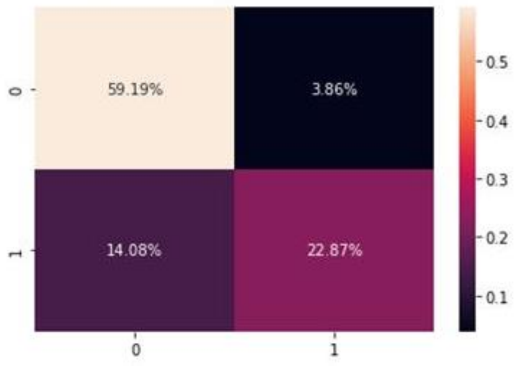
```
In [23]: model_rfc = RandomForestClassifier()
         model_rfc.fit(X_train, y_train)
         pred_rfc = model_rfc.predict(X_test)
         acc_rfc = accuracy_score(pred_rfc, y_test)
         print ('Test Accuracy : {:.2f}%'.format(acc_rfc*100))
         print(classification_report(pred_rfc, y_test))

         Test Accuracy : 95.34%
                       precision    recall  f1-score   support

                    0       0.99      0.94      0.96     23847
                    1       0.89      0.98      0.93     11970

             accuracy                           0.95     35817
            macro avg       0.94      0.96      0.95     35817
         weighted avg       0.96      0.95      0.95     35817
```

AdaBoost Classifier:



```
In [24]: model_adaB = AdaBoostClassifier(learning_rate=0.5)
         model_adaB.fit(X_train, y_train)
         pred_adaB = model_adaB.predict(X_test)
         acc_adaB = accuracy_score(y_test, pred_adaB)
         print ('Test Accuracy : {:.2f}%'.format(acc_adaB*100))
         print(classification_report(pred_adaB, y_test))

         Test Accuracy : 82.06%
                       precision    recall  f1-score   support

                    0       0.94      0.81      0.87     26243
                    1       0.62      0.86      0.72      9574

             accuracy                           0.82     35817
            macro avg       0.78      0.83      0.79     35817
         weighted avg       0.85      0.82      0.83     35817
```

```
In [25]: cf_matrix_adaB = confusion_matrix(y_test, pred_adaB)
         sns.heatmap(cf_matrix_adaB/np.sum(cf_matrix_adaB), annot=True,fmt= '0.2%')

Out[25]: <AxesSubplot:>
```

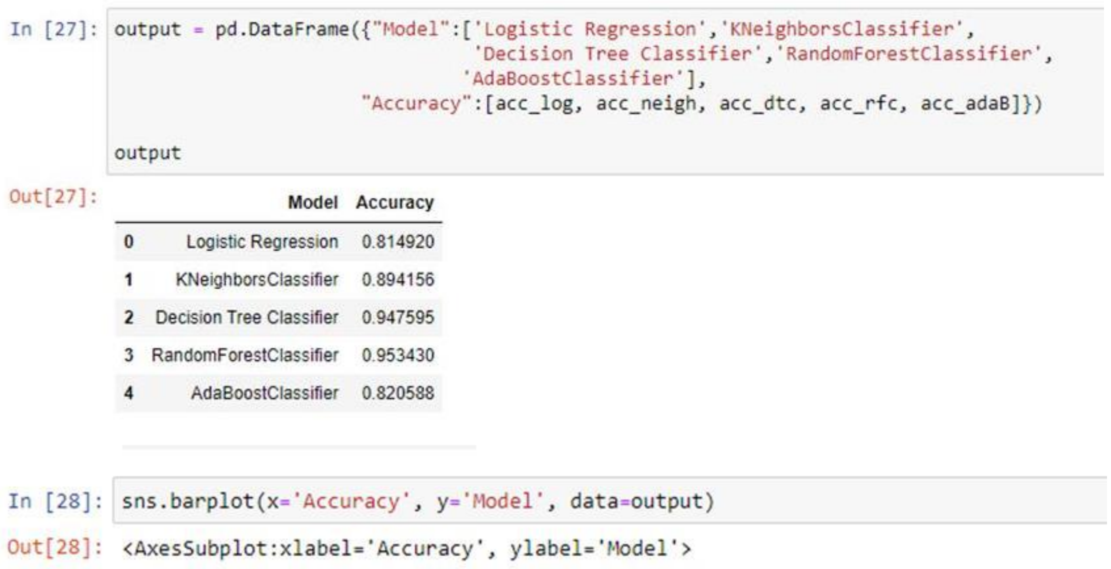


K Nearest Neighbors (KNN) Algorithm:

```
In [26]: model_neigh = KNeighborsClassifier()
         model_neigh.fit(X_train, y_train)
         pred_neigh = model_neigh.predict(X_test)
         acc_neigh = accuracy_score(y_test, pred_neigh)
         print ('Test Accuracy : {:.2f}%'.format(acc_neigh*100))
         print(classification_report(pred_neigh, y_test))

         Test Accuracy : 89.42%
                       precision    recall  f1-score   support

                    0       0.96      0.88      0.92     24752
                    1       0.77      0.93      0.84     11065

             accuracy                           0.89     35817
            macro avg       0.87      0.90      0.88     35817
         weighted avg       0.91      0.89      0.90     35817
```

Accuracy Comparison:

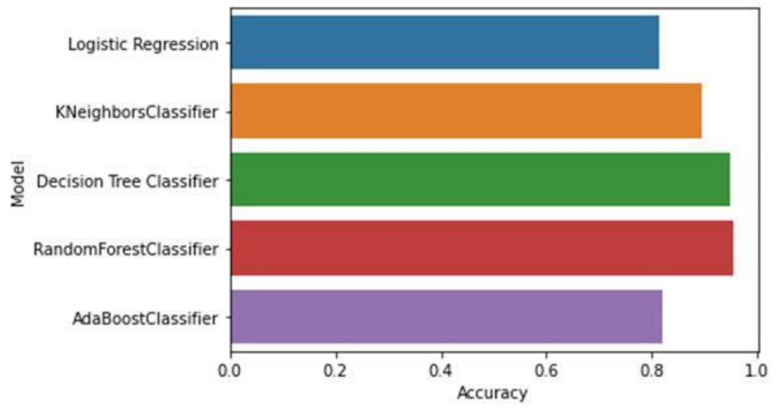
The performance of each algorithm depends on the size and complexity of the dataset, the specific characteristics and attributes of the items, and the quality and quantity of user data. Ultimately, the accuracy of any prediction algorithm depends on many factors, so it is important to evaluate the performance of each algorithm in the specific context under development. This may involve experimenting with different algorithms and continuously monitoring and optimizing the system based on user feedback and other performance metrics.



```
In [27]: output = pd.DataFrame({"Model":['Logistic Regression','KNeighborsClassifier',
                                         'Decision Tree Classifier','RandomForestClassifier',
                                        'AdaBoostClassifier'],
                               "Accuracy":[acc_log, acc_neigh, acc_dtc, acc_rfc, acc_adaB]})

         output

Out[27]:
```

	Model	Accuracy
0	Logistic Regression	0.814920
1	KNeighborsClassifier	0.894156
2	Decision Tree Classifier	0.947595
3	RandomForestClassifier	0.953430
4	AdaBoostClassifier	0.820588

```
In [28]: sns.barplot(x='Accuracy', y='Model', data=output)

Out[28]: <AxesSubplot:xlabel='Accuracy', ylabel='Model'>
```



6. CODE

```
# Importing the libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

#Importing the dataset
df = pd.read_csv('C:/Users/SNEHA/Desktop/Sem 4 Project/hotel_bookings.csv')
df.head()
#Checking for null values
df.isnull().sum()
df.children.value_counts()

#Dropping unwanted columns
df = df.drop('company',axis=1)
df['agent'] = df['agent'].fillna(0)
df['children'] = df['children'].fillna(0)
df['country'] = df['country'].fillna('Unknown')
df.isnull().sum()
```

```
#Exploratory Data Analysis
counts = df.arrival_date_month.value_counts()
counts.sort_index(inplace=True)
sns.barplot(x = counts.index, y = counts)
plt.ylabel('counts')
plt.xticks(rotation=45);
counts = df.arrival_date_year.value_counts()
counts.sort_index(inplace=True)
sns.barplot(x = counts.index, y = counts)
plt.xlabel('arrival_date_year')
plt.ylabel('counts')
counts = df.is_canceled.value_counts()
counts.sort_index(inplace=True)
sns.barplot(x = counts.index, y = counts)
plt.xlabel('is_canceled')
plt.ylabel('counts')


df_not_canceled = df[df['is_canceled'] == 0]
plt.subplots(figsize=(7,5))
sns.countplot(x='arrival_date_year', hue='hotel',  data=df_not_canceled)
sns.barplot(y="is_canceled", x="deposit_type", data=df)
sns.countplot(data=df, x = 'reserved_room_type')
plt.show()
char = df.select_dtypes(include='object')
for i in char:
    print(i , df[i].nunique())

#Dropping more columns and applying LabelEncoder
df.drop(['days_in_waiting_list', 'arrival_date_year', 'arrival_date_year',
'assigned_room_type', 'booking_changes', 'reservation_status', 'country',
'days_in_waiting_list'], axis = 1, inplace=True)
```

```
le=LabelEncoder()
df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date'])
df['year'] = df['reservation_status_date'].dt.year
df['month'] = df['reservation_status_date'].dt.month
df['day'] = df['reservation_status_date'].dt.day
df.drop(['reservation_status_date','arrival_date_month'] , axis = 1, inplace = True)

a = df.select_dtypes(object).columns
cat_list = []
for i in a:
    print (i, df[i].nunique())
    cat_list.append(i)
for i in cat_list:
    df[i] = le.fit_transform(df[i])
df['year'] = le.fit_transform(df['year'])
df['month'] = le.fit_transform(df['month'])
df['day'] = le.fit_transform(df['day'])

#Loading the dataset for traning and testing

y = df['is_canceled']
X = df.drop('is_canceled', axis = 1)
X_train, X_test, y_train, y_test = train_test_split (X,y,random_state=46,test_size=0.3)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#Building Logistic Regression model
log_reg = LogisticRegression()
log_reg.fit(X_train,y_train)
pred_log = log_reg.predict(X_test)
```

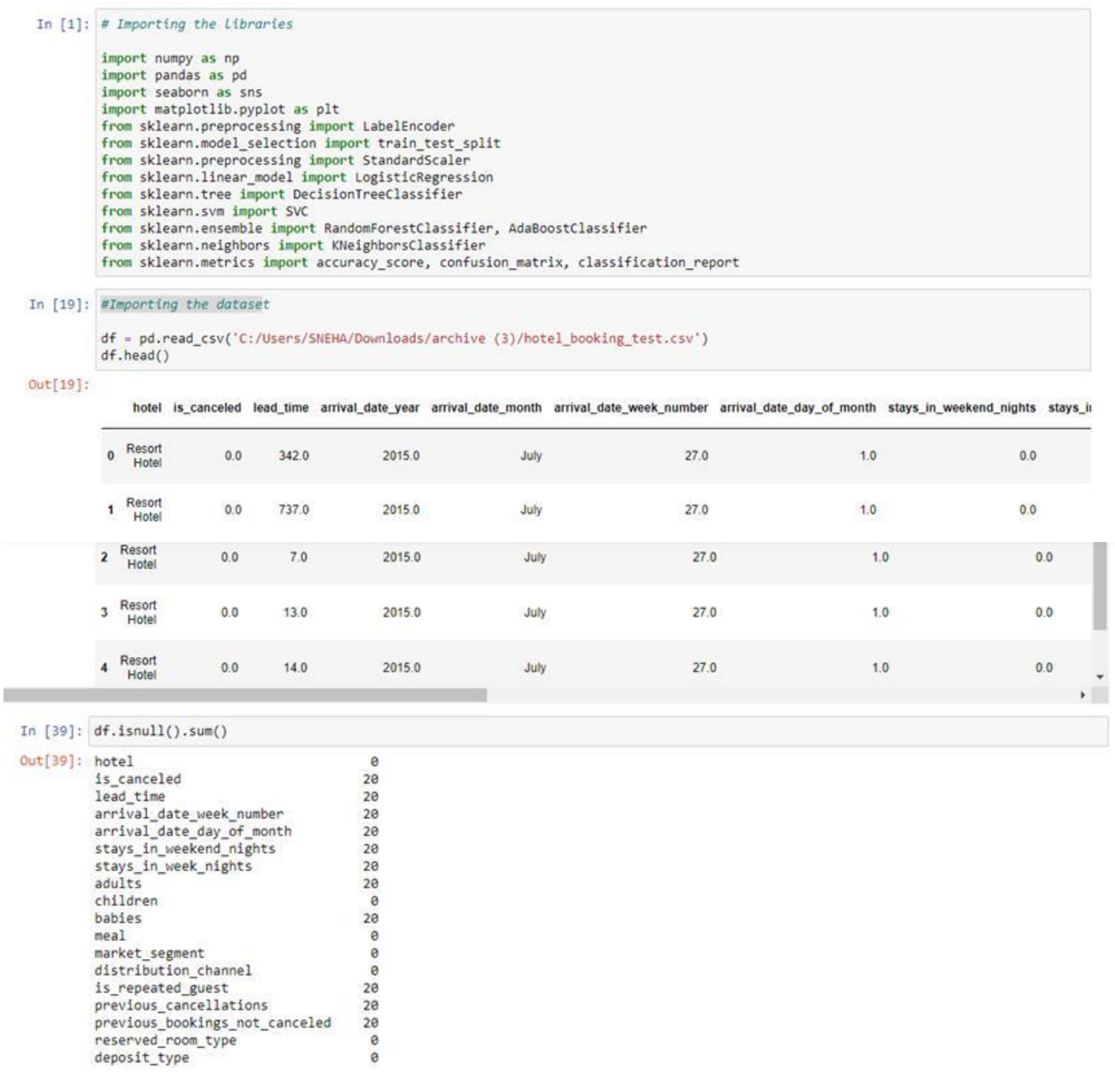
```
acc_log= accuracy_score(pred_log,y_test)
print ('Test Accuracy : {:.2f}%'.format(acc_log*100))
print(classification_report(y_test,pred_log))



#Building Decision Tree Classifier
model_dtc = DecisionTreeClassifier()
model_dtc.fit(X_train, y_train)
y_pred_dtc = model_dtc.predict(X_test)
acc_dtc = accuracy_score(y_test, y_pred_dtc)
print ('Test Accuracy : {:.2f}%'.format(acc_dtc*100))
print(classification_report(y_test, y_pred_dtc))

#Building Random Forest Classifier
model_rfc = RandomForestClassifier()
model_rfc.fit(X_train, y_train)
pred_rfc = model_rfc.predict(X_test)
acc_rfc = accuracy_score(pred_rfc, y_test)
print ('Test Accuracy : {:.2f}%'.format(acc_rfc*100))
print(classification_report(pred_rfc, y_test))

#Building AdaBoost Classifier
model_adaB = AdaBoostClassifier(learning_rate=0.5)
model_adaB.fit(X_train, y_train)
pred_adaB = model_adaB.predict(X_test)
acc_adaB = accuracy_score(y_test, pred_adaB)
print ('Test Accuracy : {:.2f}%'.format(acc_adaB*100))
print(classification_report(pred_adaB, y_test))



cf_matrix_adaB = confusion_matrix(y_test, pred_adaB)
sns.heatmap(cf_matrix_adaB/np.sum(cf_matrix_adaB), annot=True,fmt= '0.2%')
```

```
#Building KNN Classifier
model_neigh = KNeighborsClassifier()
model_neigh.fit(X_train, y_train)
pred_neigh = model_neigh.predict(X_test)
acc_neigh = accuracy_score(y_test, pred_neigh)
print ('Test Accuracy : {:.2f}%'.format(acc_neigh*100))
print(classification_report(pred_neigh, y_test))

#Comparing model accuracy
output = pd.DataFrame({"Model":['Logistic Regression','KNeighborsClassifier',
                        'Decision Tree Classifier','RandomForestClassifier',
                       'AdaBoostClassifier'],
                 "Accuracy":[acc_log, acc_neigh, acc_dtc, acc_rfc, acc_adaB]}
Output
sns.barplot(x='Accuracy', y='Model', data=output)
```

7. TESTING

Test Case Id	Test Case Description	Expected Result	Status
UT_001	Test the algorithm's ability to handle large amounts of data	Algorithm should be able to handle large amounts of data without slowing down or crashing	Pass
UT_002	Test the algorithm's ability to handle unexpected input	Algorithm should gracefully handle unexpected or incorrect input	Pass
UT_003	Test the combination between the algorithm and the machine learning model	Algorithm should be able to utilize the machine learning model to make accurate predictions	Pass
UT_004	Test the combination between the algorithm and the data preprocessing module	Algorithm should be able to receive preprocessed data from the data preprocessing module to make accurate predictions	Pass

8. SCREENSHOTS OF TEST CASES



```
In [1]: # Importing the Libraries

        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import LabelEncoder
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.svm import SVC
        from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
In [19]: #Importing the dataset

         df = pd.read_csv('C:/Users/SNEHA/Downloads/archive (3)/hotel_booking_test.csv')
         df.head()
```

Out[19]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_i
0	Resort Hotel	0.0	342.0	2015.0	July	27.0	1.0	0.0	
1	Resort Hotel	0.0	737.0	2015.0	July	27.0	1.0	0.0	
2	Resort Hotel	0.0	7.0	2015.0	July	27.0	1.0	0.0	
3	Resort Hotel	0.0	13.0	2015.0	July	27.0	1.0	0.0	
4	Resort Hotel	0.0	14.0	2015.0	July	27.0	1.0	0.0	

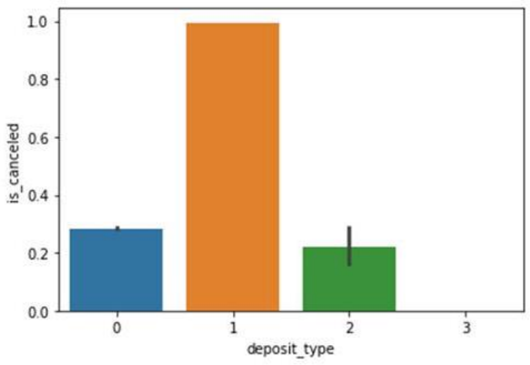
```
In [39]: df.isnull().sum()

Out[39]: hotel                              0
         is_canceled                       20
         lead_time                         20
         arrival_date_week_number          20
         arrival_date_day_of_month         20
         stays_in_weekend_nights           20
         stays_in_week_nights              20
         adults                            20
         children                           0
         babies                            20
         meal                               0
         market_segment                     0
         distribution_channel               0
         is_repeated_guest                 20
         previous_cancellations            20
         previous_bookings_not_canceled    20
         reserved_room_type                 0
         deposit_type                       0
```

```
In [21]: df.children.value_counts()

Out[21]: 0.0     110717
         1.0       4858
         2.0       3645
         3.0         76
         10.0         1
         Name: children, dtype: int64
```

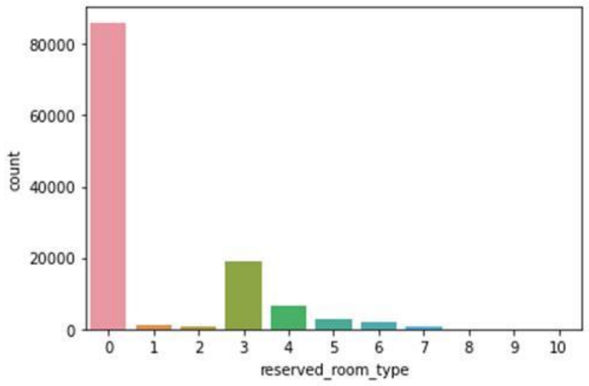
```
In [47]: df['agent'] = df['agent'].fillna(0)
         df['children'] = df['children'].fillna(0)
         df['is_repeated_guest'] = df['is_repeated_guest'].fillna(0)
         df['previous_cancellations'] = df['previous_cancellations'].fillna(0)
         df['previous_bookings_not_canceled'] = df['previous_bookings_not_canceled'].fillna(0)
         df['babies'] = df['babies'].fillna(0)
         df['stays_in_week_nights'] = df['stays_in_week_nights'].fillna(0)
         df['stays_in_weekend_nights'] = df['stays_in_weekend_nights'].fillna(0)
         df['arrival_date_day_of_month'] = df['arrival_date_day_of_month'].fillna(0)
         df['arrival_date_week_number'] = df['arrival_date_week_number'].fillna(0)
         df['lead_time'] = df['lead_time'].fillna(0)
         df['is_canceled'] = df['is_canceled'].fillna(0)
         df['adults'] = df['adults'].fillna(0)
```

```
In [49]: df.isnull().sum().sum()

Out[49]: 0
```

```
In [54]: sns.barplot(y="is_canceled", x="deposit_type", data=df)

Out[54]: <AxesSubplot:xlabel='deposit_type', ylabel='is_canceled'>
```



```
In [55]: sns.countplot(data=df, x = 'reserved_room_type')
         plt.show()
```



```
In [56]: char = df.select_dtypes(include='object')
         for i in char:
             print(i , df[i].nunique())
```

```
In [58]: for i in cat_list:
             df[i] = le.fit_transform(df[i])
         df['year'] = le.fit_transform(df['year'])
         df['month'] = le.fit_transform(df['month'])
         df['day'] = le.fit_transform(df['day'])
```

```
In [59]: y = df['is_canceled']
         X = df.drop('is_canceled', axis = 1)
         X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=46,test_size=0.3)
```

```
In [60]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

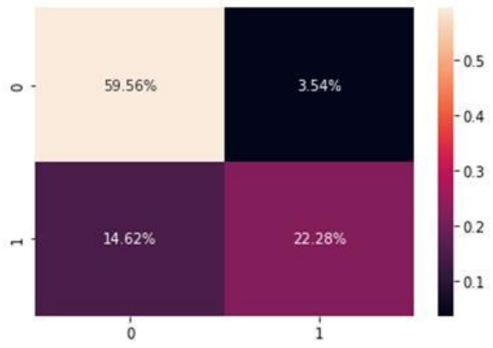
```
In [61]: model_dtc = DecisionTreeClassifier()
         model_dtc.fit(X_train, y_train)
         y_pred_dtc = model_dtc.predict(X_test)
         acc_dtc = accuracy_score(y_test, y_pred_dtc)
         print ('Test Accuracy : {:.2f}%'.format(acc_dtc*100))
         print(classification_report(y_test, y_pred_dtc))
```

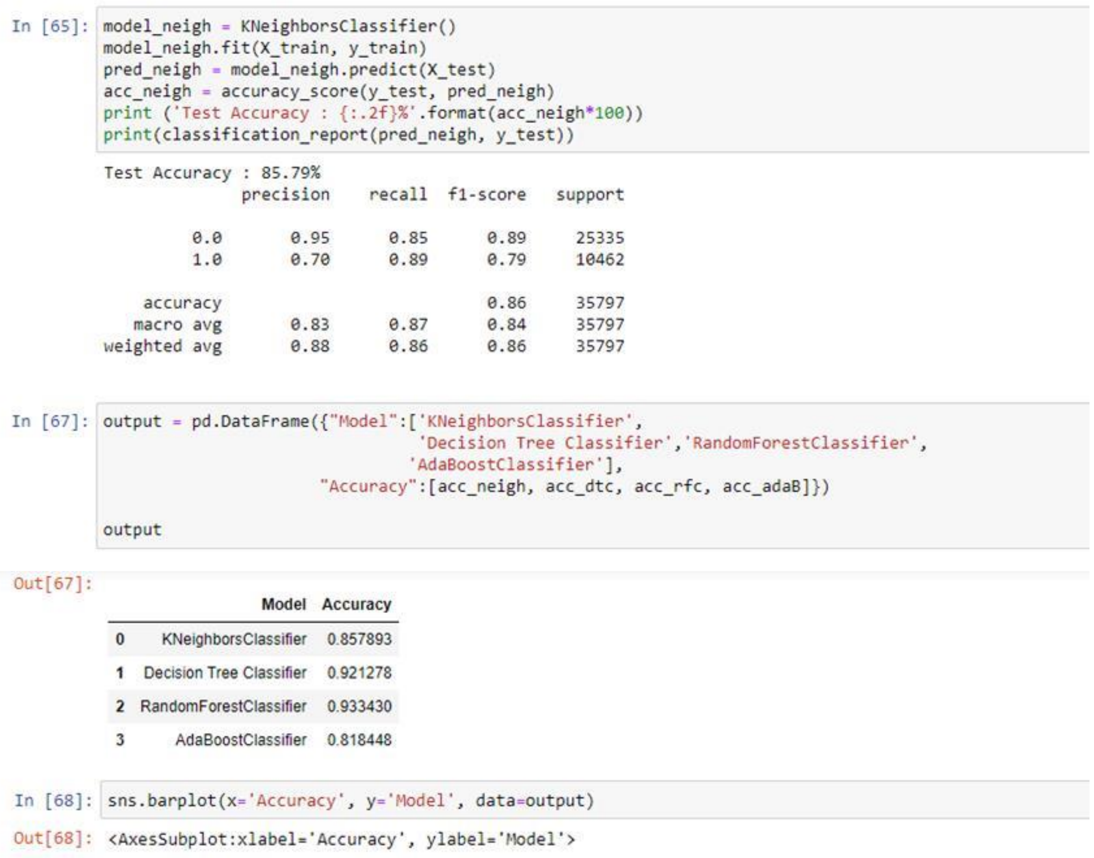
```
Test Accuracy : 92.13%
              precision    recall  f1-score   support

         0.0       0.94      0.94      0.94     22588
         1.0       0.89      0.89      0.89     13209

    accuracy                           0.92     35797
   macro avg       0.92      0.92      0.92     35797
weighted avg       0.92      0.92      0.92     35797
```

```
In [62]: model_rfc = RandomForestClassifier()
         model_rfc.fit(X_train, y_train)
         pred_rfc = model_rfc.predict(X_test)
         acc_rfc = accuracy_score(pred_rfc, y_test)
         print ('Test Accuracy : {:.2f}%'.format(acc_rfc*100))
         print(classification_report(pred_rfc, y_test))

         Test Accuracy : 93.34%
                       precision    recall  f1-score   support

                  0.0       0.99      0.91      0.95     24371
                  1.0       0.84      0.97      0.90     11426

             accuracy                           0.93     35797
            macro avg       0.91      0.94      0.93     35797
         weighted avg       0.94      0.93      0.93     35797
```

```
In [63]: model_adaB = AdaBoostClassifier(learning_rate=0.5)
         model_adaB.fit(X_train, y_train)
         pred_adaB = model_adaB.predict(X_test)
         acc_adaB = accuracy_score(y_test, pred_adaB)
         print ('Test Accuracy : {:.2f}%'.format(acc_adaB*100))
         print(classification_report(pred_adaB, y_test))

         Test Accuracy : 81.84%
                       precision    recall  f1-score   support

                  0.0       0.94      0.80      0.87     26555
                  1.0       0.60      0.86      0.71      9242

             accuracy                           0.82     35797
            macro avg       0.77      0.83      0.79     35797
         weighted avg       0.86      0.82      0.83     35797
```

```
In [64]: cf_matrix_adaB = confusion_matrix(y_test, pred_adaB)
         sns.heatmap(cf_matrix_adaB/np.sum(cf_matrix_adaB), annot=True,fmt= '0.2%')

Out[64]: <AxesSubplot:>
```





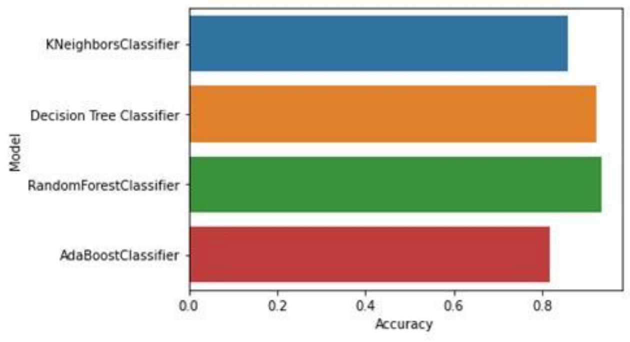
```
In [65]: model_neigh = KNeighborsClassifier()
         model_neigh.fit(X_train, y_train)
         pred_neigh = model_neigh.predict(X_test)
         acc_neigh = accuracy_score(y_test, pred_neigh)
         print ('Test Accuracy : {:.2f}%'.format(acc_neigh*100))
         print(classification_report(pred_neigh, y_test))

         Test Accuracy : 85.79%
                       precision    recall  f1-score   support

                  0.0       0.95      0.85      0.89     25335
                  1.0       0.70      0.89      0.79     10462

             accuracy                           0.86     35797
            macro avg       0.83      0.87      0.84     35797
         weighted avg       0.88      0.86      0.86     35797
```

```
In [67]: output = pd.DataFrame({"Model":['KNeighborsClassifier',
                                         'Decision Tree Classifier','RandomForestClassifier',
                                        'AdaBoostClassifier'],
                              "Accuracy":[acc_neigh, acc_dtc, acc_rfc, acc_adaB]})

         output
```

Out[67]:

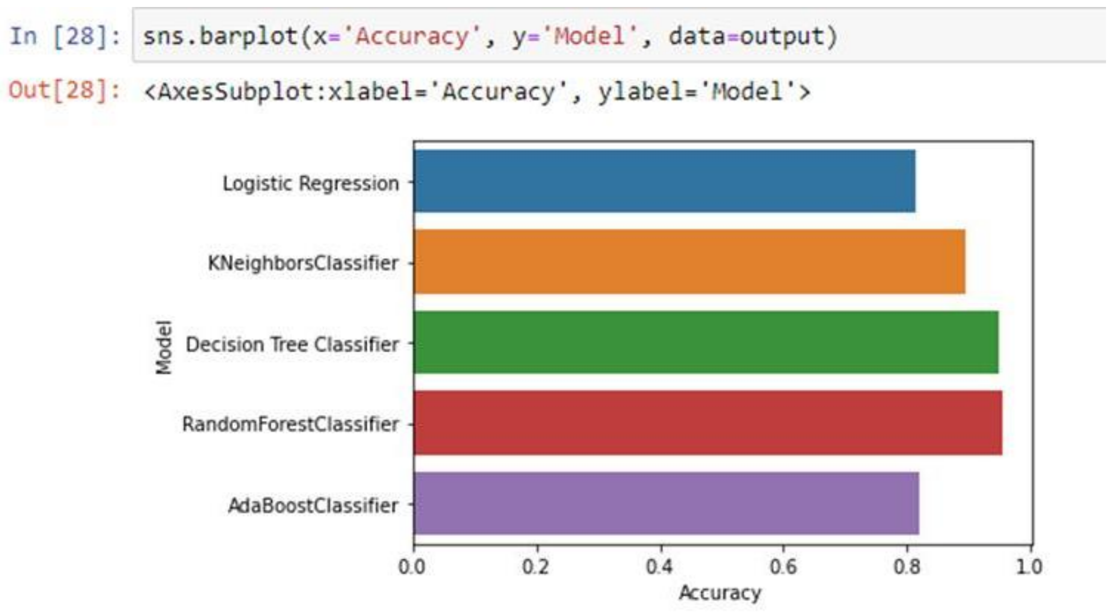
	Model	Accuracy
0	KNeighborsClassifier	0.857893
1	Decision Tree Classifier	0.921278
2	RandomForestClassifier	0.933430
3	AdaBoostClassifier	0.818448

```
In [68]: sns.barplot(x='Accuracy', y='Model', data=output)
Out[68]: <AxesSubplot:xlabel='Accuracy', ylabel='Model'>
```



Thus, for the test data as well, the Random Forest Classifier models works the best with the highest accuracy of 93% whereas KNN has an accuracy of 85%, Decision tree has an accuracy of 92% and AdaBoost Classifier has an accuracy of 81%.

Accuracy Comparison Graph



```
In [28]: sns.barplot(x='Accuracy', y='Model', data=output)

Out[28]: <AxesSubplot:xlabel='Accuracy', ylabel='Model'>
```

This graph demonstrates that Random Forest Classifier is the best algorithm among all the algorithms with accuracy of 95%.

9. CONCLUSION

The given dataset is a supervised classification dataset. It holds booking information for a city hotel and a resort hotel with information such as how and when the booking was made, the length of passengers' stay with the number of parking slots available, the number of adults, children, and babies. The Logistic regression, K-Nearest Neighbour, Decision Tree, Random Forest algorithms are used to handle this supervised classification model. Among these four machine learning algorithms, Random forest and Decision trees perform well with respect to accuracy. The random forest classifier model is developed based on preprocessed and analyzed hotel_bookings.csv dataset and augmented features. The original dataset is consists of 119390 data entries with 31994 duplicated data. It is consist of 32 features of:

The preprocessed data used for the model development is consist of 87010 dataset and 14 features. The significant features used are based on the outcome from the multi-attribute analysis using the spearman method. Based on the finding, the features are selected based on their rank, reducing the dataset dimensions. In this case, the default parameters for the classifier model without optimization.

The base model results here show acceptable outcome and performance. The confusion matrix shows a good number of True Positive compared to False Positive with acceptable False Negative and True Negative. It is also to note here the amount of True Positive is significantly greater than True Negative. Intuitively, this is true as confirmed_booking is usually > cancelled_booking for the dataset. This results in an acceptable model accuracy of 77%. The model accuracy is greatly influenced by the reduction of dataset (removed duplicate) which can be either positively improve the model or cause overfitting.

10. FUTURE SCOPE

It would be advantageous to implement the algorithm in a real environment and evaluate its performance on a larger scale with real users. This can provide valuable insight into the practicality and effectiveness of algorithms in real-world contexts. This research employed data from four resort hotels that use the same PMS, which raises some questions that further research could help.

Consequently, additional research, with different PMS data, additional hotels, and additional types of hotels could contribute to a better understanding of the topic of booking cancellation prediction. Further research could also make use of features from additional data sources, such as weather information, competitive intelligence (prices and social reputation), or currency exchange rates, to improve model performance and measure the influence of these features in booking cancellations. Finally, deployment of these predictive models in a production environment, in hotels, with the purpose of executing machine learning models, could contribute to measure the effect of having previous knowledge of which bookings have a high cancellation probability.

11. BIBLIOGRAPHY

1. N. Antonio, A. Almeida, L. Nunes, Predicting hotel bookings cancellation with a machine learning classification model, in: Proc. 16th IEEE Int. Conf. Mach. Learn. Appl., IEEE, Cancun, Mexico, 2017: pp. 1049–1054. doi:10.1109/ICMLA.2017.00-11.

2. A Grouping Hotel Recommender System Based on Deep Learning and Sentiment Analysis, Fatemeh Abbasi, Ameneh Khadivar, Mohsen Yazdinejad, 2019.

3. Studying the cancellation behaviour of the guests, Markku Vieru , 2016

4. Andrew, W. P., Cranage, D. A., and Lee, C. K. (1990). Forecasting hotel occupancy rates with time series models: An empirical analysis. Hospitality Research Journal, 14(2):173–182.

5. Friedman, J., Hastie, T., and Tibshirani, R. (2001). The elements of statistical learning, volume 1. Springer series in statistics New York.

6. GlobalData (2017). How can artificial intelligence, machine learning, and big data boost efficiency in the industry? Case study: Machine learning in the hotel industry